Intelligent Service Robotics (2018) 11:25–39
https://doi.org/10.1007/s11370-017-0243-8

ORIGINAL RESEARCH PAPER

Routing-based navigation of dense mobile robots

Rahul Kala[1]

Received: 4 December 2016 / Accepted: 22 November 2017 / Published online: 4 December 2017

Abstract
In the future, many teams of robots will navigate in home or office environments, similar to dense crowds operating currently in different scenarios. The paper aims to route a large number of robots so as to avoid build-up of congestions, similar to the problem of route planning of traffic systems. In this paper, first probabilistic roadmap approach is used to get a roadmap for online motion planning of robots. A graph search-based technique is used for motion planning. In the literature, typically the search algorithms consider only the static obstacles during this stage, which results in too many robots being scheduled on popular/shorter routes. The algorithm used here therefore penalizes roadmap edges that lie in regions with large robot densities so as to judiciously route the robots. This planning is done continuously to adapt the path to changing robotic densities. The search returns a deliberative trajectory to act as a guide for the navigation of the robot. A point at a distant of the deliberative path becomes the immediate goal of the reactive system. A 'centre of area'-based reactive navigation technique is used to reactively avoid robots and other dynamic obstacles. In order to avoid two robots blocking each other and causing a deadlock, a deadlock avoidance scheme is designed that detects deadlocks, makes the robots wait for a random time and then allows them to make a few random steps. Experimental results show efficient navigation of a large number of robots. Further, routing results in effectively managing the robot densities so as to enable an efficient navigation.

Keywords Motion planning · Navigation · Swarm robotics · Multi-agent systems · Routing · Probabilistic roadmap

1 Introduction

The problem of motion planning for multiple mobile robots [1,2] is to make the robots go from their pre-defined source to their pre-defined goals, such that none of the robots collides with any obstacles, and no two robots collide with each other. Let $S = \{S_i\}$ be the sources of the robots and $G = \{G_i\}$ be the goals of the robots. Let $C^{\text{free}}_{\text{static}}$ be the free configuration space of the robots considering only the static obstacles and no other robot. Similarly let C_{static} be the total configuration space. $C^{\text{obs}}_{\text{static}}$ is the obstacle-prone configuration space, $C^{\text{obs}}_{\text{static}} = C_{\text{static}} \backslash C^{\text{free}}_{\text{static}}$. The problem is to compute trajectories $T = \{\tau_i\}$ of all robots, such that each robot starts from its source (S_i), ends at its goal (G_i) and does not collide with any static obstacle or other robots on the way.

The problem is usually solved by using a hybrid of deliberative and reactive planning [3,4]. The deliberative planner is good for avoiding trap situations and caters to the needs of optimality and completeness of the approach and however requires a large computation time. Reactive planning on the other hand is very good to avoid dynamic and suddenly appearing obstacles, while moving in real time and however often makes the robot trapped and does not guarantee optimality and completeness. A hybrid of both techniques is hence a good choice. Numerous schemes have been proposed for using a mixture of deliberative and reactive planning. A popular way is to make the deliberative planner give an approximate path of the robot τ'_i, in many occasions not considering the other robots and dynamic obstacles. The path so produced is used as a guide by the reactive planner. A goal g_i is selected in τ'_i, which the reactive planner aims to attain. The goal g_i is taken at a fixed distance from the current position of the robot and is moved as the robot travels. Eventually the robot reaches the goal G_i.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11370-017-0243-8) contains supplementary material, which is available to authorized users.

✉ Rahul Kala
rkala001@gmail.com
http://rkala.in/

1 Robotics and Artificial Intelligence Laboratory, Indian Institute of Information Technology, Allahabad, India

In the deliberative hierarchy, one needs to constantly plan (and re-plan in this case) the trajectory of every robot. A good way of doing so is by using the probabilistic roadmap (PRM) [5,6] approach which samples out the free configuration space by taking samples in $C^{\text{free}}_{\text{static}}$ which become the vertices of a roadmap $R\langle V, E\rangle$. The sampling strategy decides the selection of samples, which become the vertices. It is common to have more samples near obstacle boundaries and in narrow corridors. The neighbouring vertices are connected by an edge $\langle v_i, v_j\rangle$, if a trajectory between v_i and v_j can be found by using a local planning algorithm. A popular mechanism is to use straight line connections as the local planning algorithm that is an edge $\langle v_i, v_j\rangle$ is added if $\lambda v_i + (1-\lambda)v_j \in C^{\text{free}}_{\text{static}} \forall 0 \leq \lambda \leq 1$. Stronger edge connection mechanisms are often used for connecting difficult regions.

The use of deliberative planning is to avoid the robot getting stuck at trap locations, by constantly guiding the robot nearer to the trap-free deliberative path. However, the mobile robots also act as dynamic obstacles to each other and create trap situations even in simple maps. Since the number of robots is assumed to be high, such deadlocks are un-avoidable. Even two robots, desiring to go in opposing directions, can cause a deadlock to each other. In the case of humans, such deadlocks are resolved by communication and common sense; a suitable algorithm is proposed in this work to do the same for robots.

If the deliberative planning is done without considering the other robots, it is evident that many robots will be scheduled in the same areas at the same time, increasing congestion in the same areas. An inspiration is taken from the problem of routing in transportation systems [7,8], wherein it is fundamentally known that if all the vehicles take the shortest or the fastest route, the congestion at the popular roads will be excessively large. Congestions cause none of the vehicles to move for prolonged times, thus significantly increasing the time of travel. A popular method is hence to avoid the build-up of congestions. The studies in transportation systems cannot be used to solve the same problem in robot motion planning, as the transportation is a discrete graph of vertices and edges in which all vehicles travel outbound or inbound; however, the robot motion planning problem is continuous in nature wherein any robot can behave in any manner. It may be argued that the roadmap of the PRM can be interpreted as a discrete graph of the transportation systems. However, the physical motion of the robots is not and should not be restricted to travelling on the roadmap edges alone, thus wasting most of the part of the space unnecessarily causing congestions.

The approach adopted here is biologically inspired. While walking, we sometimes see a big crowd of people around, and accordingly, we re-plan our way to avoid the high density of people. Similarly, the proposed approach considers the current density of robots while making the route of a robot. Only the immediate density is considered. It is assumed that all robots operate independently and without communication, and hence knowing the future densities is not possible. Anticipation is very hard in case of such open-ended maps.

Since the densities change with time, it is important to constantly re-plan the deliberative plan. Every re-plan is as per the changed density map of the environment. For the same reasons PRM is chosen as the choice of deliberative algorithm, which has an offline roadmap construction phase and an online query computation phase.

The main contributions of the work are: (i) development of a simulation framework for simulating the motion of a large number of robots. (ii) Inspired from the problem of routing in transportation systems, design of a routing strategy so as to avoid excessive build-up of robots at a place. Unlike transportation systems, the routing happens in continuous spaces rather than discrete road network graphs. (iii) Design of a deadlock avoidance scheme that can continuously eliminate deadlocks between different groups of robots. (iv) Continuous re-planning for adapting the changes in the robot flows.

This paper is organized as follows: Sect. 2 very briefly lists some relevant related works. The algorithm framework is presented in Sect. 3. The approach involves construction of an offline roadmap (Sect. 3.1), which is used numerous times for online planning. The problem of routing for judicious distribution of robots is done (Sect. 3.2) to eliminate the build-up of high density of robots. The robot moves using a reactive planner (Sect. 3.3) based on the deliberative path in a hybrid architecture (Sect. 3.4). Deadlocks may happen, which are detected and resolved (Sect. 3.5). The results are given in Sect. 4. The conclusion remarks are given in Sect. 5.

2 Related works

In the literature much less work has been done in the context of motion of a large number of dense mobile robots. Although much work exists in the related problems of motion of robotic swarms [9] wherein the swarm moves collectively and is thus homogeneous in contrast to the problem of motion of multiple robots. Similarly there are works in crowd simulation which simulate a large number of crowds, and aim at imitation of the human motion by virtual agents. The work in the domains of both robot swarms and crowd simulation has not considered the problem of global congestion avoidance by active re-routing of the agents. Even though some of the tools account for density, the density avoidance is similar to obstacle avoidance and is not done so as to route the agents to avoid subsequent build-up of congestions on popular routes. Some of the popular techniques are discussed.

Pelechano et al. [10] modelled for different crowd behaviours including avoiding obstacles and people, avoiding walls, queuing, pushing, panic propagation, impatience, etc., hence allowing for the motion of people. Musse and Thalmann [11] presented different behavioural models like flocking, following, space availability, safe wandering for the motion of crowds, which could also be controlled by scripts and events. Curtis et al. [12] used behavioural finite state machine modelling to identify goals of crowds, thereafter computing and adapting the path. Similarly Gu and Deng [13] aimed to make the simulation process more realistic and modelled different styles of agent motion. The simulator aimed at increasing the style diversity among the agents. Kountouriotis et al. [14] specifically concentrated on the reactive navigation of the agents to see the emergence of group behaviours by a microscopic modelling technique. The authors modelled different forces of goal seeking, aggregation, obstacle avoidance, friction and body compression.

In another related work Han et al. [15] demonstrated the notion of routing of humans for the specific task of evacuation. The route choice depended upon the factors of density, congestion at exit, length of the route and the distance between the pedestrian and route. A potential field modelling was used for the actual navigation of the robot. The simulations were based upon the dispersion behaviour which inhibits interaction between groups going in opposite direction, which is the main crux of the proposed approach. Golas et al. [16] further used anticipation to predict the temporal densities of crowds, which was taken as a probabilistic distribution. The authors used the same for long-range collision avoidance. The anticipation does result in spreading the crowd and hence congestion avoidance; however, the method is computationally expensive that restricts its use to the level of re-routing for congestion control. The proposed algorithm hence heuristically handles the same problem.

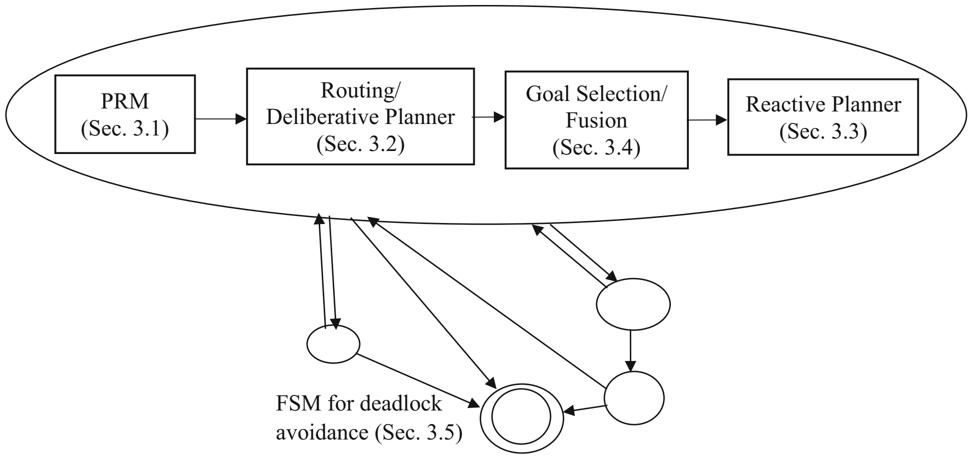
A lot of work is also done in hierarchical planning of the robots. Cowlagi and Tsiotras [17] used A^* algorithm at two levels for planning a robot. The finer level A^* algorithm computed the transition costs that were used by the coarser-level A^* algorithm. In another vein Sgorbissa and Zaccaria [18] used Voronoi graphs for planning a robot at the coarser level, while the actual motion was done using the potential field approach. The authors avoided deadlock by identifying scenarios called roaming trails, wherein the robot was unable to move as per the deliberative plan. Similarly, Chang and Yamamoto [19] made a robot operate using a Voronoi which was built as the robot navigated for exploration, with the specific motion of the robot guided by potential field for obstacle avoidance. Roadmaps are also used with the presence of multiple robots. Yao and Gupta [20] used multiple robots to sense the environment and to collectively make a roadmap in a distributed manner. Clark [21] also used a single query PRM wherein multiple robots communicated with each other to make a complete roadmap. Chai and Su [22] presented a mechanism of coordinating the motion of multiple robots. At the centralized level the problem was motion of a group towards a goal by forming a certain shape. In case of infeasibility, the shape constraints were changed. Density regularization and congestion avoidance is clearly not the theme of any of the hierarchical approaches. The concentration is only on the reduction in computation time, because of which the other agents are mostly ignored during deliberative planning.

The reactive planning used in this paper is inspired from the work of Alvarez-Sanchez et al. [23] who performed a circular scan of the environment to get the clearest direction of motion of the robot, and the work of Sezer and Gokasan [24] who found the largest gap ahead of the robot and used the same for moving the robot, considering the non-holonomic constraints. Similarly Kim and Kwon [25] used vanishing point of the obstacle and assessment of angle of the obstacle in order to make the robot navigate towards the goal while avoiding collisions on the way. In this paper the ideas have been adapted to the navigation of small dense robots and knit in the overall framework. Further, the reactive paradigm considers both the clearance and path length as a metric for navigation decisions.

Another popular method of reactive planning is velocity obstacles [26], wherein infeasible immediate velocities of the robots are eliminated from the set of possible velocities, from which an immediate navigation velocity is selected based on heuristics. Hybrid reciprocal velocity obstacle [27] method eliminates the problem of oscillations in such an approach by assigning the potentially colliding robots with opposing corrections. Similarly Rashid et al. [28] proposed to use orientations as well in the selection of velocities. Unlike velocity obstacles, the choice of method for navigation is geometric in nature considering that the robots are capable of and will largely change their speeds as they travel, which for a large number of robots can be better modelled using the geometric approach. Further, the proposed approach allows for insertion of new obstacles of arbitrary shapes during navigation, which is more realistic.

In another related reactive navigation approach Karagoz et al. [29] presented the mechanism by which multiple disc-shaped robots can go to their goals by following a potential-based approach. The authors identified goal selection schemes that guarantee convergence. The behaviours were restricted to goal seeking. Similarly Zhong et al. [30] learnt a model for navigation from observations of crowd motion in videos using genetic programming. Using the learnt model, the authors could predict motion even to scenarios not directly seen during learning. Patil et al. [31] constructed navigation fields to decide the path of an agent, while using local collision checking algorithms for the motion of the agent in pursuit of the goal. The papers tackle the reactive paradigm of the problem only. This paper puts in the notion of deliberation for planning complex maps and

Fig. 1 Algorithm methodology



deliberation for collision avoidance in addition to these basic behaviours.

The proposed approach samples out the given configuration space using PRM. Effective sampling of the configuration space is another problem for which a lot of active research exists. Hybridization of samplers creates possibilities for the different samplers to complement each other, thus resulting in a compact roadmap that can be quickly generated. Hsu et al. [32] used different samplers in the generation of the roadmap and used performance measures of these samplers to adaptively vary the contributions of the individual samplers. In another related work Rodrıguez et al. [33] classified different areas of the configuration space based on obstacle density and used appropriate samplers for each region. Morales et al. [34] determined the difficulty of a region based on its edge connectivity and sampled so as to enhance the connectivity.

As evident from the literature, the context of routing of robots is absent from the literature of multi-robot motion planning, while the literature around dense simulation of robots, each with a different source and goal pair, is slim.

3 Algorithm

Given a free configuration space $C^{\text{free}}_{\text{static}}$, the sources $S = \{S_i\}$ and goals $G = \{G_i\}$ of all robots, the problem is to compute the trajectories $T = \{\tau_i\}$ of the robots such that they start from the source $\tau_i(t_i) = S_i$, end at the goal, $\tau_i(t) = G_i \quad \forall t \geq t_i^G$. Here t_i^G is the time when the robot i reaches its goal. The robots should not collide with any static obstacles, nor should the robot collide with each other, that is Eq. (1)

$$\left(\tau_i(t) \in C^{\text{free}}_{\text{static}}\right) \wedge \left(R\left(\tau_i\left(t\right)\right) \cap R\left(\tau_j\left(t\right)\right) = \phi\right) \quad \forall t, i, j, i \neq j \tag{1}$$

here $R(\cdot)$ is the function that maps the robot's configuration space to the set of points in the workspace. Two robots are non-colliding if their set of intersection points in the workspace is empty.

The space is continuous in nature that is converted into a graph by the use of a probabilistic roadmap technique. The graph so-generated can be used for searching a solution to the goal for every robot by a graph search algorithm, and the problem is called as routing. The search attempts to minimize path length and minimize congestion. The path may still not be navigable by the robot due to the presence of other robots and robot constraints, for which a reactive technique of navigation is designed. The deliberative technique using routing guides the reactive navigator, creating a fused deliberative and reactive navigation system. Finally, the complete scheme may still have deadlocks for which a FSM-based deadlock avoidance system is used. The discussed methodology forms one of the states of the FSM, while transitions occur on detection and for preventing deadlocks. The complete approach is summarized in Fig. 1.

3.1 Probabilistic roadmap construction

The first part of the problem is to generate the roadmap $G\langle V, E\rangle$. The approach is adapted from an earlier work of the author [35]. For the generation of the vertices, a hybrid of different sampling techniques is used. The first sampler used is a narrow corridor sampler that generates a sample inside obstacles, promotes the sample to the obstacle-free configuration space and uses a bridge test [36] to see whether the sample is in a narrow corridor. The second sampler is an obstacle-based sampler [37] that generates a sample in the obstacle-prone configuration space and promotes it to the obstacle-free configuration space. The last sampler is a uniform sampler.

For connecting the vertices, initially edge connections with the neighbouring k nearest vertices are made. Subsequently, a hybrid edge connecting strategy is used. The first strategy attempts to join disjoint set of vertices so as to get a path between all points of the configuration space. The disjoint set is selected stochastically based on the distance

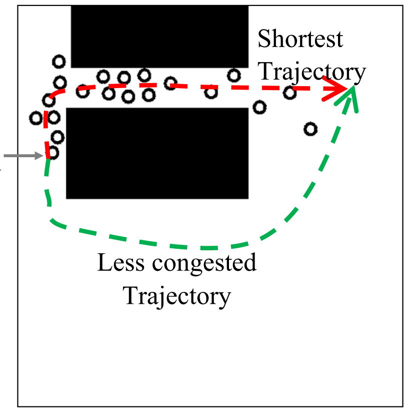


Fig. 2 Avoiding congestion. Robot A should take the less congested trajectory rather than the shortest trajectory so as to avoid congestion and reach its destination earlier. This is achieved by actively routing the robot on a congestion centric metric. The robot continuously searches for shorter and less congested paths

between the sets, closer disjoint sets being more desirable. The participating vertices are also selected stochastically based on their distance. Connection attempts involve adding a new vertex intermediate between the participating vertices and making edge connections if possible. The second strategy of edge connection selects a leaf node in the roadmap and grows it in a random direction like expansive space trees. The third strategy creates a lower-dimensional memory hash and encourages generation of vertices in cells with lower vertex density. The sample is used to grow the roadmap in a rapidly exploring random trees style expansion [38,39]. The last connection strategy makes a random expansion of the roadmap in a rapidly exploring random tree style expansion.

The roadmap is used for the online planning of the robots. A temporary roadmap $R'\langle V', E'\rangle$ is created with all vertices and edges of the roadmap $R'\langle V', E'\rangle$. Source and goal of the robot queried are added as additional vertices. These are connected to the nearest k vertices, if a straight line connection is possible. A^* algorithm [40] is used to compute the trajectory τ_i' of the robot, the trajectory used to guide the reactive planner.

3.2 Routing

Consider the scenario shown in Fig. 2. If the robot A was human, trying to go from the left of the corridor to the right, he/she would have the sense to take the longer route as too many people are already going through the corridor, making the route congested. Hence, weights of the roadmap $R\langle V, E\rangle$ need to be modified to get the same attributes in the motion of the robots. The proposal is to add a penalty term to the costs of the edges, so as to avoid taking routes with high density.

Let $\rho(x, y)$ be the density at the point (x, y). Considering that microscopic variables are maintained, while the intention is to compute the macroscopic variable density, let the density be recorded at an observation window of size $w \times w$. The density is the ratio of the area occupied by the robots to the total area of the observation window, given by Eq. (2).

$$\rho(x, y) = \frac{\left(\text{count}_{x-w/2 \le \tau_{i,t}^X \le x+w/2 \wedge y-w/2 \le \tau_{i,t}^Y \le y+w/2}(i)\right)\pi r^2}{\text{area}(W_{\text{static}}^{\text{free}} \cap [x - w/2, x + w/2] \times [y - w/2, y + w/2])} \tag{2}$$

$\tau_{i,t}$ is the current position of robot i of radius r with X component as $\tau_{i,t}^X$ and Y component as $\tau_{i,t}^Y$. $W_{\text{static}}^{\text{free}}$ is the free workspace of the robot; the component in the observation window is considered. $\text{count}_C(x)$ is used to denote count of all x that satisfy the condition c. The observation window is trimmed if it is outside the navigable area.

Consider the straight line edge $e\langle v_i, v_j\rangle$ from v_i to v_j. A point at a distance of l from v_i, denoted by $e(l)$, is given by Eq. (3). Here $||.||$ is the Euclidian norm, which is the distance between the vertices. The cost of the edge is given by Eq. (4).

$$e(l) = v_i + l\frac{v_j - v_i}{\|v_j - v_i\|} \tag{3}$$

$$w\left(e\left\langle v_i, v_j\right\rangle\right) = \int_{l=0}^{l=\|v_i - v_j\|} (1 + \alpha \cdot \rho(e(l))).dl \tag{4}$$

here α is the constant relating density to penalty. $1+\alpha \cdot \rho(\cdot)$ is the penalty term. If the visible density is large, the penalty term increases, making it less likely for the robot to select the path, and vice versa. If α is very large, the algorithm starts preferring robot-free paths, if it can find one. If α is 0, the algorithm reduces to a shortest path search.

3.3 Reactive planning

Considering the instantaneous position of robot as q at time t, the task of the reactive planner is to determine its position at the next time step $t + \Delta$ based on some control input $u_i^{\text{reactive}}\langle v_i^{\text{reactive}}, \omega_i^{\text{reactive}}\rangle$. Let g_i be the goal given to the reactive planner. Let θ be the direction of the vector $g_i - q$, which is the desired angle of motion not accounting for the other robots. The notations are shown in Fig. 3a. So that the robot does not get stuck, the immediate motion of the robot is restricted between $\theta - \gamma$ and $\theta + \gamma$, where γ is the allowable window of deviation from going straight to the goal. Let $d(\nu)$ be a restricted proximity sensor that senses distances to a maximum of L in the direction ν. The approach used in this paper is entirely based on the geometric assessment of the obstacles as perceived by the sensors. Another popular method is the social potential field method [41,42] wherein the different obstacles exert virtual forces that are modelled using social relations between robots and obstacles. The choice here is primarily based on the intuition that

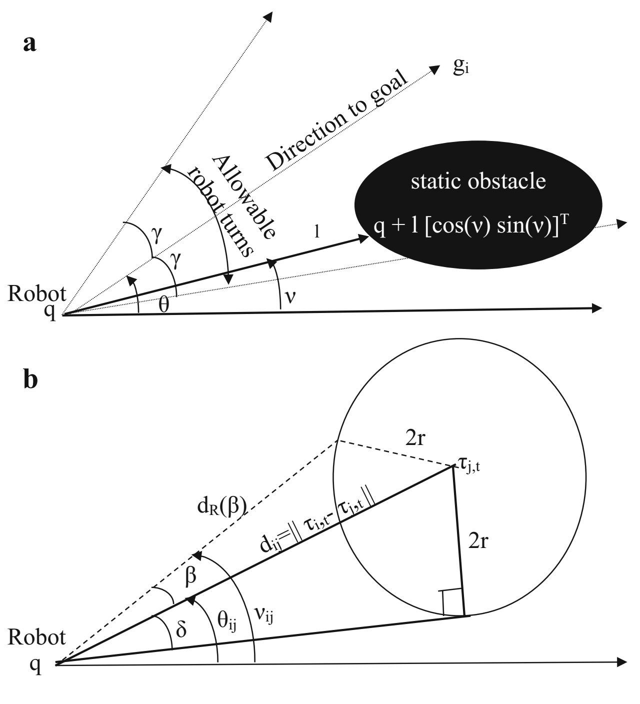


Fig. 3 Computation of distance **a** between robots and a static obstacle, **b** between two robots

the humans tend to place themselves right in the middle of the available region, while the social potential approach does not necessarily model this.

In order to make the distance function, one must calculate the distance at direction ν in $C^{\text{free}}_{\text{static}}$ by traversing through the line in the same direction, given by Eq. (5). The terms are illustrated in Fig. 3a.

$$d_{\text{obs}}(\nu) = \min_{l \leq L, q + l[\cos(\nu) \;\; \sin(\nu)]^T \in C^{\text{free}}_{\text{static}}} (l) \tag{5}$$

here $\min_c(x)$ denotes minimize x subjected to the condition c.

Similarly the minimum distance with the other robots must be computed. For this consider the configuration space of the robot centred at q, with only the other robots centred at q_j. This can be achieved by taking a point robot centred at q and the other robots with sum of the radii of the two robots. Consider Fig. 3b, using the law of cosines, the distance is given by Eq. (6).

$$d_R(\beta) = d_{ij}\cos(\beta) - \sqrt{d_{ij}^2 \cos^2(\beta) - \left(d_{ij}^2 - 4r^2\right)},$$
$$\beta = \nu - \theta_{ij}, \quad -\delta \leq \beta \leq \delta \tag{6}$$

here d_{ij} is the distance between q_j and q, and θ_{ij} is the angle between q_j and q. δ is the limiting angle for which the formula holds, given when the line becomes a tangent to the circle, $\delta = \sin^{-1}(2r/d_{ij})$. The notation $-\delta \leq \beta \leq \delta$ is

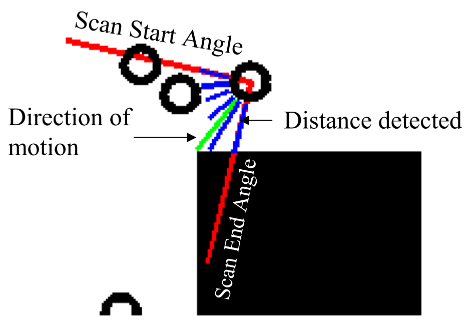


Fig. 4 Computation of angle of movement. Red lines show the cone in which distance is scanned. Blue line shows the detected minimum distances. The distances are assuming double radius of the other robots, such that the robot as a whole is avoided. Green line shows the direction of maximum distance used for motion

an abuse of notation as it numerically does not account for the circular nature of the angles. The distance function can hence be given by Eq. (7). The equation simply states that the minimum distance from obstacle and robot is considered, subjected to a threshold of L.

$$d(\nu) = \min\left(d_{\text{obs}}(\nu), d_R(\nu), L\right) \tag{7}$$

The direction of travel (ϕ) of the robot is taken as the direction which maximizes the distance. In case of a tie, the largest gap is taken. Further of all the regions that host the largest distance, the middle point of the region hosting the largest gap is taken. The general concept is shown in Fig. 4. The angular speed $\omega_i^{\text{reactive}}$ is set so as to obtain the direction of travel ϕ, given by Eq. (8). Equation (8) is again an abuse of notation as the angle differences need to maintain the circular property, which is not shown in the equation.

$$\omega_i^{\text{reactive}} = \begin{cases} (\phi - \theta)/\Delta & \omega_{\min} \leq (\phi - \theta)/\Delta \leq \omega_{\max} \\ \omega_{\min} & (\phi - \theta)/\Delta > \omega_{\max} \\ \omega_{\max} & (\phi - \theta)/\Delta < \omega_{\min} \end{cases} \tag{8}$$

The linear speed setting is given by Eq. (9).

$$v_i^{\text{reactive}} = \begin{cases} v_i^{\max} & d(\phi)/\Delta \geq 2v_i^{\max} \\ d(\phi)/2\Delta & \text{otherwise} \end{cases} \tag{9}$$

The speed setting enables the robot to travel with the maximum speed, as far as possible. Once the robot is closer to a static obstacle or another robot, the speed is slowly reduced, such that the speed slowly tends to 0 as the robot approaches the robot or obstacle.

The navigation of the robot is assumed to be using an infinite acceleration model in both the linear and angular speeds. So the linear and angular speeds can be instantaneously set. A small additional check is performed to ensure that the selected speed and steering makes the new position feasible, failing which the speed is iteratively reduced till the settings are feasible. The calculations shown above are not made on the radius r as shown in the equations; instead, a small comfortable distance of ε is kept so as to eliminate two robots nearly touching each other or the obstacle boundary, if possible. The pseudo-code of the approach is given as Algorithm 1.

Algorithm 1: Reactive Navigation

```
for $\nu$ from $\theta-\gamma$ to $\theta+\gamma$ in some resolution
        calculate $d(\nu)$ using equations (5), (6) and (7)
$d_{max} \leftarrow max_\nu(d(\nu))$
$(\phi_i^1, \phi_i^2) : d(\phi_i) = d_{max} \forall\ \phi_i^1 \le \phi_i \le \phi_i^2$
$\phi \leftarrow (\phi_i^1 + \phi_i^2)/2$
calculate $\omega_i^{reactive}$ from equation (8)
calculate $v_i^{reactive}$ from equation (9)
return $v_i^{reactive}$, $\omega_i^{reactive}$
```

3.4 Fusion of deliberative and reactive planning

The deliberative planner gives an indicative trajectory τ_i' for the robot to follow. The reactive planner needs a goal and continuously moves the robot so as to move towards the goal avoiding the obstacles. The deliberative planning is resolution optimal that is the planning generates the optimal path subjected to some resolution, and resolution complete that is the planning guarantees to find a solution if one exists subjected to some resolution. However, the approach is very computationally expensive and cannot be used for real-time navigation of the robot. The approach cannot be used to handle sudden changes in the map. If changes in the map are reported, after some time the path can be adapted by re-planning. In contrast, the reactive planner neither guarantees optimality nor completeness. The algorithm parameters can be tuned so as to get some local optimality that enables the robot to maintain optimal distances from the obstacles around. However, the robot may not follow the optimal route towards the goal in a complex obstacle grid. In fact, most of the times the robot may get trapped in an obstacle framework and may never be able to react to the goal. The reactive planner is very fast in computation. It can therefore react to any suddenly appearing obstacle or sudden changes in the map.

It must be noted that the completeness and optimality notions here are defined for a static scenario only and in the absence of other robots. For the case of multiple robots these properties can only be ascertained by centralized planners which work by considering the joint states and joint actions of all robots, and is not an option when the number of robots is so high due to the exponential complexity. Decentralized planners do not guarantee optimality, but tend to be near-optimal subjected to the validity of the heuristic used, which here is penalization due to density.

The fusion of deliberative and reactive planning enables moving the robot as a result of the added advantages of

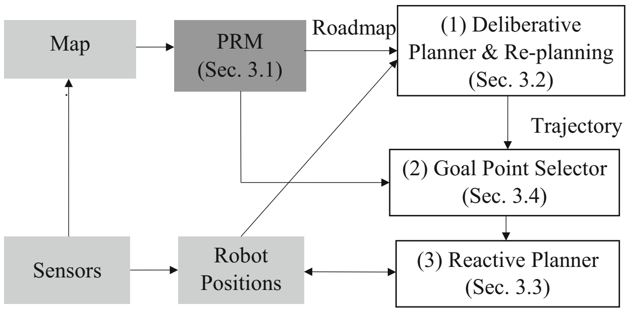

Map
PRM (Sec. 3.1)
Roadmap
(1) Deliberative Planner & Re-planning (Sec. 3.2)
Trajectory
(2) Goal Point Selector (Sec. 3.4)
Sensors
Robot Positions
(3) Reactive Planner (Sec. 3.3)

Fig. 5 Fusion of deliberative and reactive planning. The non-shaded boxes form the main logic flow of the program, while the shaded boxes denote the standard robot operations

the two methodologies. Here a master–slave architecture of fusion is used. The deliberative planner works at the master level and is responsible for deliberative planning and re-planning of the robot. The slave reactive planner looks at the obstacles and robots around and makes a move. The integration is based on the goal point. The deliberative planner supplies a goal point g_i, which is taken as the goal of the reactive planner. The fusion module looks at the deliberative trajectory and searches for the goal point g_i. The goal of the reactive planner, g_i, is taken as the point at a distance of more than D from the current position p_i. This is given by Eq. (10).

$$g_i = \begin{cases} \tau_i'(s) : \int_{t=0}^{s} \tau_i'(t) \mathrm{d}t = D & \int \tau_i'(t) \cdot \mathrm{d}t \le D \\ G_i & \int \tau_i'(t) \cdot \mathrm{d}t > D \end{cases} \quad (10)$$

The fusion may be visualized as the reactive planner being tempted by showing a goal point, asking the reactive planner to move towards it. As the robot moves towards the goal, the goal moves following the deliberative trajectory so as to be at a distance of D from the robot. The fusion architecture is shown in Fig. 5.

If the distance D is taken too large, the robot essentially moves using the reactive planner with negligible contribution of the deliberative planner, and vice versa. The pseudo-code for the approach is given by Algorithm 2.

Algorithm 2: Reactive Planner Move

```
$\tau_i' \leftarrow$ PRMQuery(q,$G_i$) (Section 3.2)
Calculate $g_i$ using equation (10)
$\theta \leftarrow$ direction($g_i - q$)
Calculate $v_i^{reactive}$, $\omega_i^{reactive}$ from Algorithm 1
Move the robot using $v_i^{reactive}$, $\omega_i^{reactive}$
```

3.5 Deadlock avoidance

The scheme so designed, running by a density-aware active deliberative re-planning and reactive planning solves most of the problems in multi-robot motion planning. The deliberative mechanisms are supposed to make the robot avoid trap

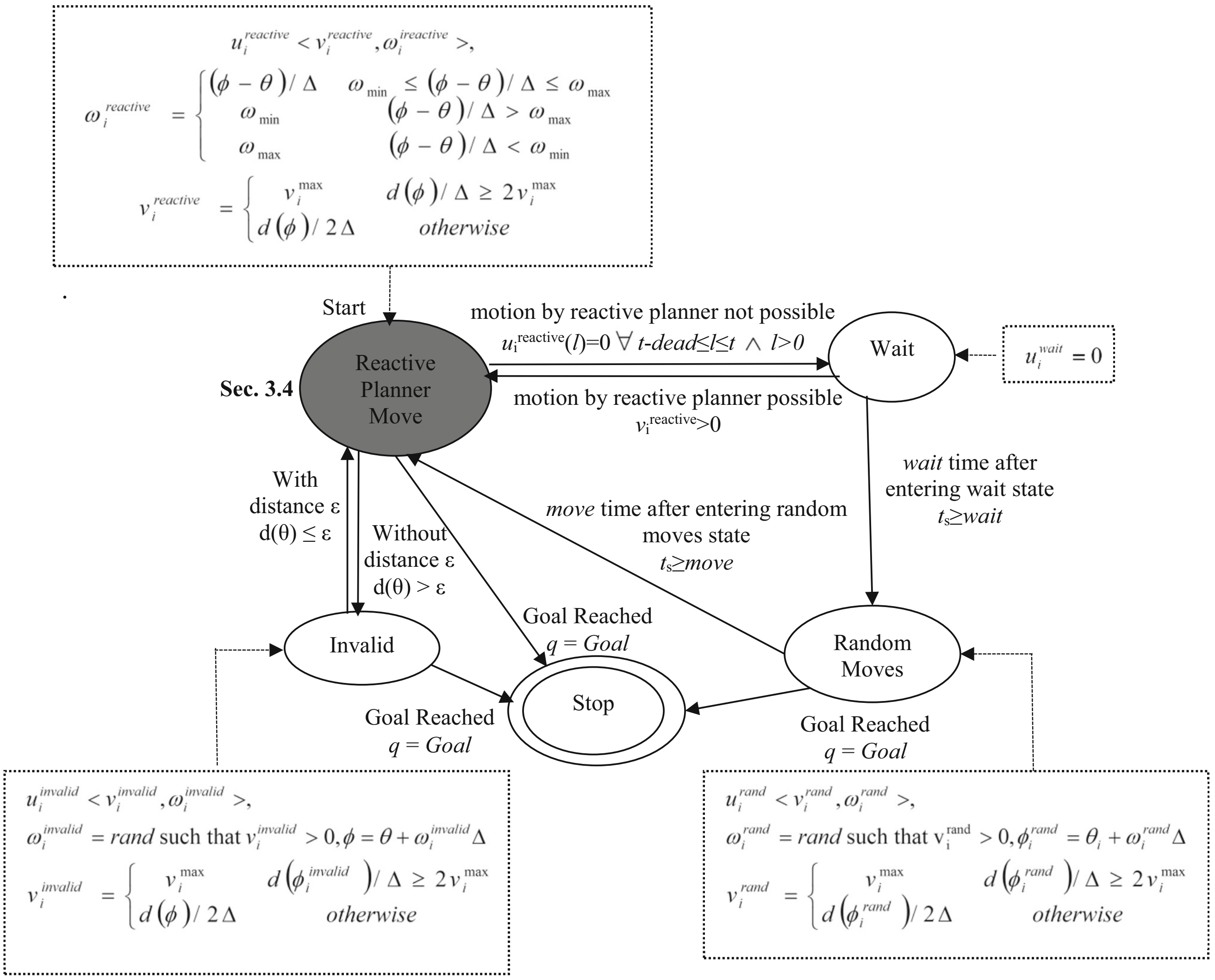

$u_i^{reactive} < v_i^{reactive}, \omega_i^{reactive} >$,
$\omega_i^{reactive} = \begin{cases} (\phi-\theta)/\Delta & \omega_{\min} \le (\phi-\theta)/\Delta \le \omega_{\max} \\ \omega_{\min} & (\phi-\theta)/\Delta > \omega_{\max} \\ \omega_{\max} & (\phi-\theta)/\Delta < \omega_{\min} \end{cases}$
$v_i^{reactive} = \begin{cases} v_i^{\max} & d(\phi)/\Delta \ge 2v_i^{\max} \\ d(\phi)/2\Delta & otherwise \end{cases}$
Start
Reactive Planner Move
Sec. 3.4
motion by reactive planner not possible
$u_i^{reactive}(l)=0\ \forall\ t\text{-}dead \le l \le t \wedge l>0$
Wait
$u_i^{wait} = 0$
motion by reactive planner possible
$v_i^{reactive}>0$
wait time after entering wait state
$t_s \ge wait$
move time after entering random moves state
$t_s \ge move$
With distance ε
$d(\theta) \le \varepsilon$
Without distance ε
$d(\theta) > \varepsilon$
Goal Reached
$q = Goal$
Invalid
Stop
Random Moves
Goal Reached
$q = Goal$
Goal Reached
$q = Goal$
$u_i^{invalid} < v_i^{invalid}, \omega_i^{invalid} >$,
$\omega_i^{invalid} = rand$ such that $v_i^{invalid} > 0, \phi = \theta + \omega_i^{invalid}\Delta$
$v_i^{invalid} = \begin{cases} v_i^{\max} & d(\phi_i^{invalid})/\Delta \ge 2v_i^{\max} \\ d(\phi)/2\Delta & otherwise \end{cases}$
$u_i^{rand} < v_i^{rand}, \omega_i^{rand} >$,
$\omega_i^{rand} = rand$ such that $v_i^{rand} > 0, \phi_i^{rand} = \theta_i + \omega_i^{rand}\Delta$
$v_i^{rand} = \begin{cases} v_i^{\max} & d(\phi_i^{rand})/\Delta \ge 2v_i^{\max} \\ d(\phi_i^{rand})/2\Delta & otherwise \end{cases}$

Fig. 6 Deadlock avoidance. The circles represent the states, the lines represent the transitions with the transition conditions written on the arrows, and the dotted boxed blocks show the continuous dynamics

situations to some extent, considering that the static obstacles are accounted for. However, the mobile robots behave as special kind of obstacles, not considered in the deliberative planning, that produce trap situations. Since there are too many other robots, the chances of producing deadlock situations are very high.

The deadlock situation is resolved using a hybrid finite state machine modelling. The modelling is given in Fig. 6. The hybrid system represents continuous dynamics of the robot inside the states, while there are a set of discrete state changes. In the problem, the states are given by Eq. (11).

$$S = \{\text{Reactive Planner Move, Wait, Invalid, Random Moves, Stop}\} \tag{11}$$

The continuous dynamics and discrete step changes in states together constitute the hybrid system. The system proposed is a timed hybrid system, wherein a timer is always maintained and can be used for continuous dynamics or state changes. At any time the robot is at either of the states $s \in S$, while starting from the initial state $s_0 =$ Reactive Planner Move. The transition from one state to the other takes place based upon the guard condition. A set of guard conditions are associated with every state, and whenever the condition becomes true, a discrete transition occurs. The guard conditions are labelled edges on the directed graph denoting the hybrid finite state machine. A transition from the state a to b occurs if the condition $T(a, b)$ evaluates to be true.

Whenever the robot is moving, it is in the moving or reactive planner move state. The continuous dynamics of the system for the state is already described by Eqs. (5–9). A deadlock is said to have occurred when the robot is not moving for *dead* units of time, excluding very minor position adjustments. The transition condition is given by Eq. (12).

$$T(\text{reactive planner move, wait}) := \Big(u_i^{\text{reactive}}(l) = 0 \forall t - dead \le l \le t \wedge l > 0\Big) \quad (12)$$

The parameter *dead* is a constant. High values result in too much time wasted to call a deadlock. Considering that a deadlock avoidance many times leads to another deadlock situation, and numerous deadlock attempts may be required to fully resolve all possible deadlocks, the undue time wasted can make the motion slow. Small values can wrongly call situations as deadlock, like motion through congested areas. Deadlock avoidance calls for making suboptimal moves, and hence, this should be avoided.

Once a deadlock is detected, the robot moves in a wait state. In this state the robot waits for a random amount of time, *wait*, subjected to a maximum of $wait^{\max}$. The waiting time is not completed if the deadlock naturally gets cleared, giving way for the robots to make some move ahead. The waiting time is important when two robots are causing each other deadlock, moving one of them clears the way for the other. So two robots can be asked to wait, and after the first takes the deadlock avoidance strategy, the way for the other is already clear to normally move. The time is kept as random so as to have a probabilistic completeness of the clearance of deadlock. Two robots causing deadlock, with only one conflicting way to go ahead for each, can indefinitely come into a deadlock if they both simultaneously start the conflicting move, realizing the conflict cancel the move, wait for the same time and re-start the conflicting move again. Waiting for too long is not suggestive, since it wastes time. Waiting for too small time can cause all conflicting robots to simultaneously make random moves as a result creating a chaos. The continuous dynamics of the state is simply no motion, given by Eq. (13). The transition from wait state is given by Eq. (14). Here t_s denotes the time spent in the specific state, obtained from the timer. However, if the deadlock naturally clears, the robot continues to move using the reactive planner. The transition condition is given by Eq. (15).

$$u_i^{\text{wait}} = 0 \quad (13)$$

$$T(\text{wait, random moves}) := (t_s \ge wait) \quad (14)$$

$$T(\text{wait, reactive planner move}) := (v_i^{\text{reactive}} > 0) \quad (15)$$

After waiting for the prescribed time, if the deadlock still does not clear, the robots make *move* number of random moves. The move is taken randomly for *move* times, subjected to a maximum of $move^{\max}$, for the same reasons. This motion is uninterrupted, since it may appear that a deadlock is clear, which may actually not be clear, landing up the robot in the same state. Moving too little may cause the robot to come in exactly the same deadlock again. Making too many random moves may make the motion of the robot as suboptimal, taking the robot too much out of course. Once the random steps are made, the robot again attempts to move normally as per the reactive system. It may still get into a deadlock, in which case all the steps are repeated. The transition condition is given by Eq. (16).

$$T(\text{random moves, reactive planner move}) := (t_s \ge move) \quad (16)$$

In order to move the robot randomly, first a random direction ϕ is chosen and then the speed in the same direction is computed as per equation (9). If moving appreciably in the direction is not possible due to feasibility, a new direction is chosen, till the robot makes a move. Speed may be reduced to below the value computed by Eq. (9) if the same is somehow not feasible. If the robot is currently not in a feasible position due to noise or invalid motion of the other robot, the random move can also add feasibility. The continuous dynamics is given by Eq. (17–18).

$$u_i^{rand}\langle v_i^{rand}, \omega_i^{rand}\rangle, \omega_i^{rand} = rand \text{ such that } v_i^{\text{rand}} > 0, \phi_i^{rand} = \theta_i + \omega_i^{rand}\Delta \quad (17)$$

$$v_i^{rand} = \begin{cases} v_i^{\max} & d\left(\phi_i^{rand}\right)/\Delta \ge 2v_i^{\max} \\ d\left(\phi_i^{rand}\right)/2\Delta & \text{otherwise} \end{cases} \quad (18)$$

A preference kept in the design is that the robots should always keep a small comfortable distance ε. If the same is not maintained due to reasons of noise or improper working of the other robots, the robot enters the invalid state, wherein random moves are made till the invalidity is resolved. This behaviour is not needed in the wait state, since a waiting robot is no threat and the moving robots should adjust instead. The behaviour is not needed in the random moves state, since any invalidity is also rectified by the state. The dynamics in this state is given by Eqs. (19–20). The transition conditions are given by Eqs. (21–22).

$$u_i^{\text{invalid}}\langle v_i^{\text{invalid}}, \omega_i^{\text{invalid}}\rangle, \omega_i^{\text{invalid}} = rand \text{ such that } v_i^{\text{invalid}} > 0, \phi = \theta + \omega_i^{\text{invalid}}\Delta \quad (19)$$

$$v_i^{\text{invalid}} = \begin{cases} v_i^{\max} & d\left(\phi_i^{\text{invalid}}\right)/\Delta \ge 2v_i^{\max} \\ d(\phi)/2\Delta & otherwise \end{cases} \quad (20)$$

$$T(\text{Reactive Planner Move, Invalid}) := (d(\theta) \le \varepsilon) \quad (21)$$

$$T(\text{Invalid, Reactive Planner Move}) := (d(\theta) > \varepsilon) \quad (22)$$

The stop state is a special state when the entire simulation run stops and therefore the state is not associated with any dynamics. The transition to the stop state happens on reaching the goal, given by Eqs. (13)–(25).

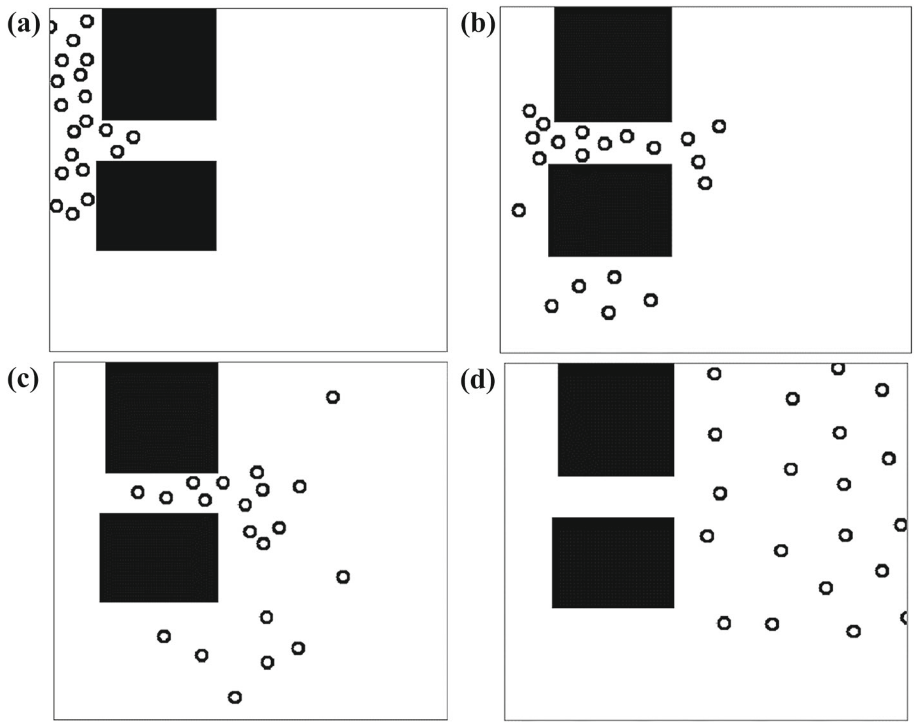


Fig. 7 Results for the first scenario

$$T(\text{Reactive Planner Move, Stop}) := (q = \text{Goal}) \quad (23)$$

$$T(\text{Invalid, Stop}) := (q = \text{Goal}) \quad (24)$$

$$T(\text{Random Moves, Stop}) := (q = \text{Goal}) \quad (25)$$

The outputs from this machine can be collected both from the discrete transitions and the continuous dynamics. In this system the state of the robot q and the control signal applied u is the output that is used for navigation.

4 Results

The approach is tested for a variety of scenarios with different number of robots. Five scenarios are presented. Each scenario presents a different challenge that increases the complexity of the overall solution. In the simulations, any of the moving entities can be taken as a human. In simulation it was ensured that no communication exists between any two entities and therefore it does not matter whether a robot is surrounded by a human or other robots; its motion will remain the same. The purpose behind the simulation at the microscopic level is to study the navigation of a robot amidst humans and other dynamic obstacles. However, in simulation it was not possible to simultaneously manually control multiple such agents, and hence a computer program had to be used to control such agents autonomously. The scenarios are presented in the following subsections.

4.1 Scenario 1

In the first scenario a narrow corridor is taken and 20 robots are generated at one end of the corridor. The robots need to travel at the other end of the corridor. Travelling through the narrow corridor is much shorter as compared to taking the longer turn around. However, because of density consciousness, many robots take the longer route and thus save from build-up of congestion. The results are shown in Fig. 7. Figure 7a is for the source positions. Figure 7b shows that while most of the robots take the shorter route in the middle of the obstacle, some of the robots instead start to take the longer route so as to avoid congestion. The complete distribution is shown in Fig. 7c wherein the robots have distributed themselves. Figure 7d shows the goal positions. The results are also presented in supplementary video. The results of all scenarios are also summarized in Table 1.

4.2 Scenario 2

The second scenario is more general, featuring 49 robots to travel from their source to the goal. The results are shown in Fig. 8. Figure 8a shows the initial positions of the robots. In Fig. 8b as the robots proceed towards the goal, they develop a little congestion. The congestion gets eventually cleared as shown in Fig. 8c. The final positions of the robots are shown in Fig. 8d.

Table 1 Metrics associated with the simulations for different scenarios

S. no.	Scenario	Simulation time (s)	Iterations	Mean straight line distance from source to goal
1	Scenario 1	8.87	266.1	327.6601
2	Scenario 2	8.67	260.1	260
3	Scenario 3	3.9	117	381.6171
4	Scenario 4	5.7	152	419.2627
5	Scenario 5	6.67	200	369.8162

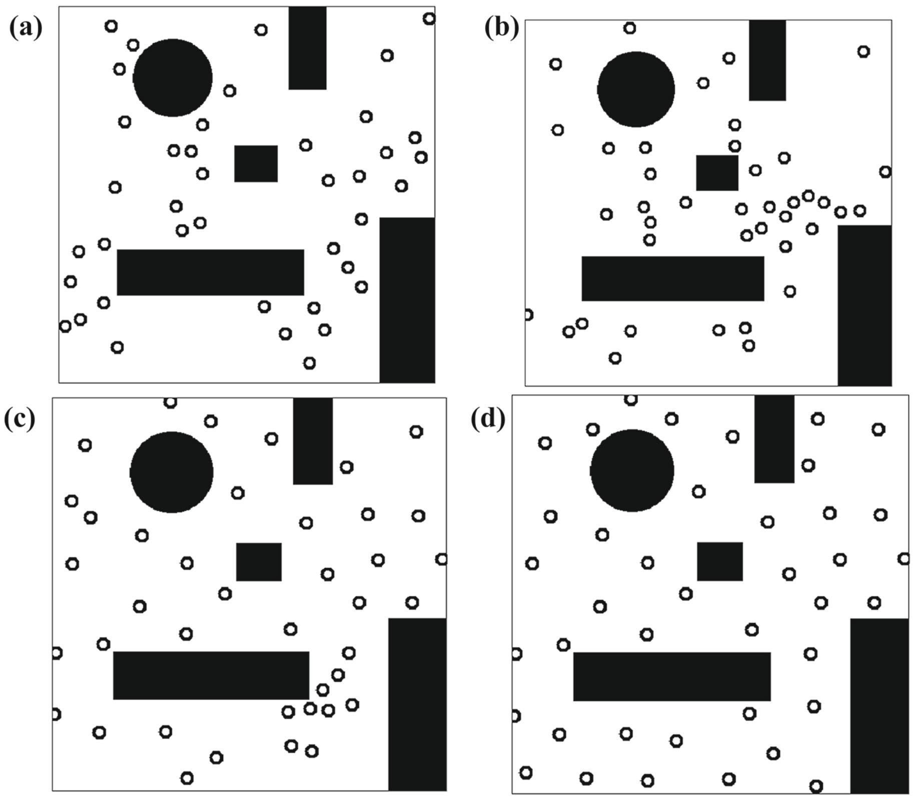


Fig. 8 Results for the second scenario

4.3 Scenario 3

For further testing, a scenario was designed so that there are multiple routing options for a group of robots. The aim was to avoid an unnecessary build-up of congestion over the shortest route area and to spread the robots in space. The results are shown in Fig. 9. Figure 9a shows the initial positions from where the robots start. Figure 9b shows the robots distributing themselves so as to reach their goal as well as to avoid congestion. Subsequently, there are more routing options available at the second column of obstacles, shown in Fig. 9c. The robots again distribute themselves so as to minimize the length of path, while still not developing any congestion. Figure 9d shows the final positions.

In the next scenario, multiple robots are generated from different corners that cross each other with an aim to meet in the middle and create congestion. The results are shown in Fig. 10 with Fig. 10a showing the initial positions. Initially, there was a build-up of robots; however, on realizing this the other robots kept away and started taking longer non-congestion-prone routes. Due to the choice of scenario, some congestion at the centre had to develop. The build-up of congestion is shown in Fig. 10b, while the robots finally into congestion are shown in Fig. 10c. Soon the robots got clear of congestion and moved to their final positions shown in Fig. 10d.

4.4 Scenario 5

To further enable a large number of robots naturally gather around at a place, another scenario was designed. Here robots from two sources had to actually cross each other, creating an unavoidable congestion in the centre with less routing options after being initially committed to a route. The results are shown in Fig. 11. Figure 11a shows the source positions. There was a build-up of robots at a narrow area as

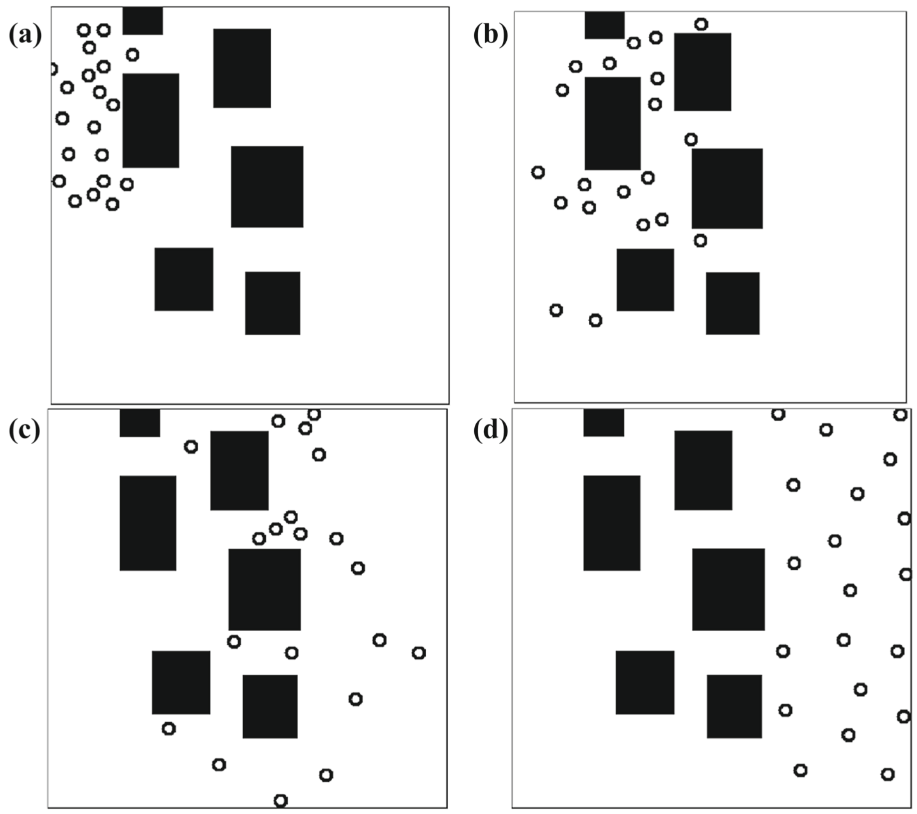


Fig. 9 Results for the third scenario

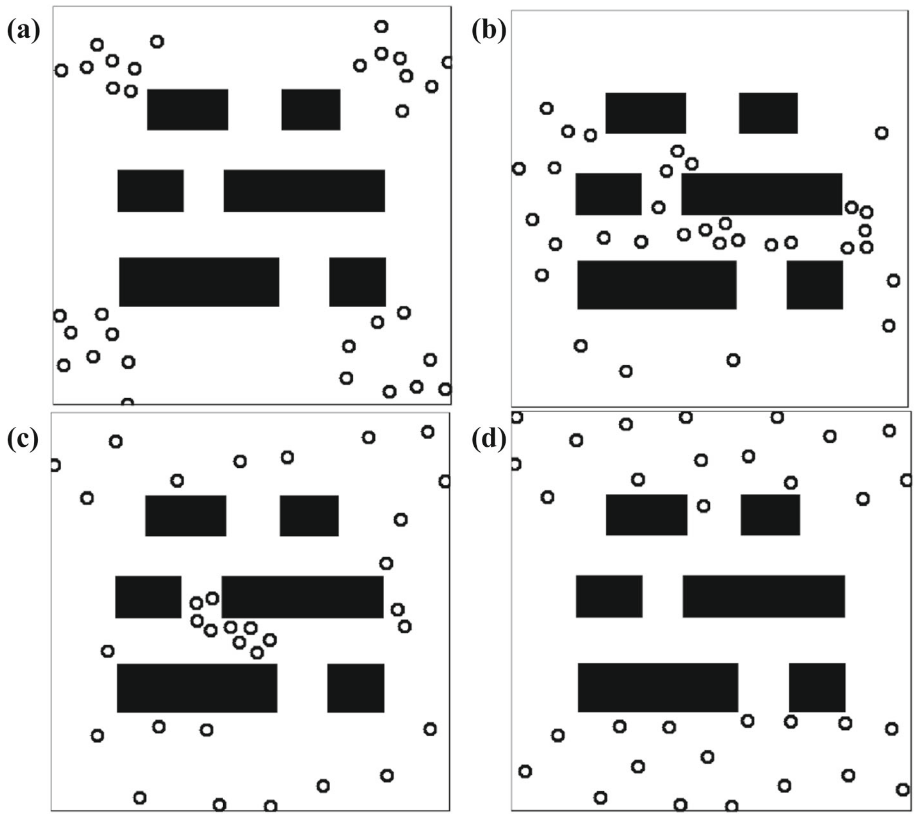


Fig. 10 Results for the fourth scenario

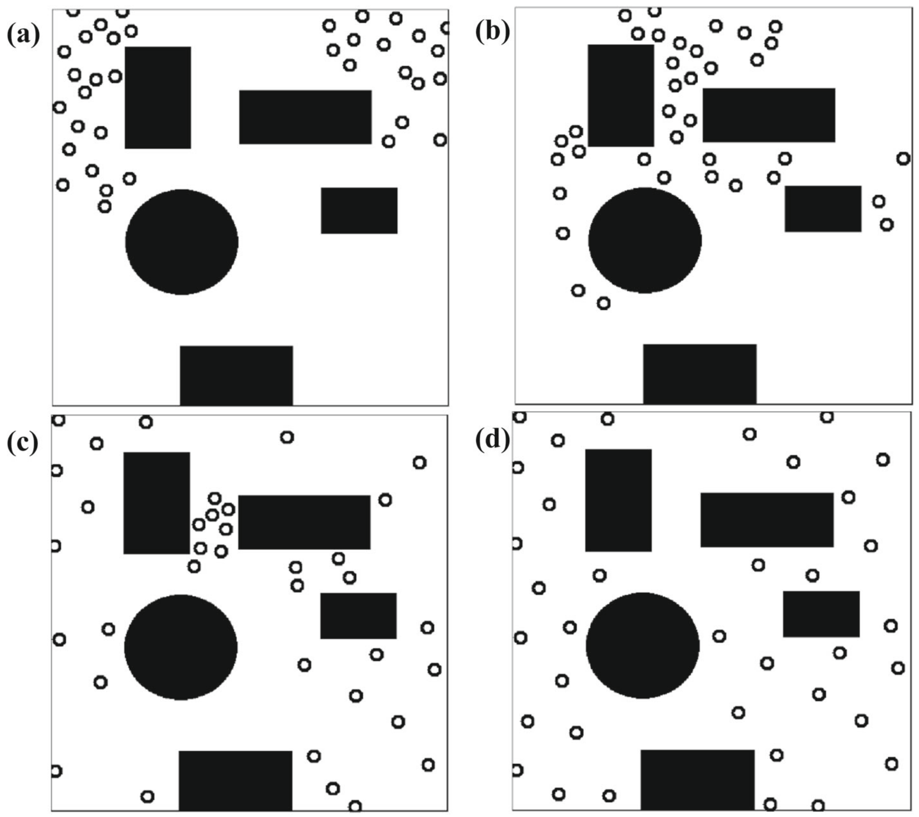


Fig. 11 Results for the fifth scenario

was intended by the simulation design, shown in Fig. 11b. However, soon thereafter the algorithm pushed the robots around to take a far longer route, while the robots at the narrow region aligned with each other to clear way as shown in Fig. 11c. The robots finally reached their goal positions shown in Fig. 11d.

4.5 Analysis of the parameter α

Routing is one of the key aspects of the approach. The routing algorithm has a major parameter α or the penalty constant. The parameter is tested for performance for the first scenario. Lower values of the parameter encourage more robots to travel through the narrow corridor, and hence, the distance travelled is small. However, due to the increased congestion inside the narrow corridor, the time taken by the robots to wait is very large. Larger values of the parameter make the robots unnecessarily conscious of the density, and hence, the robots prefer taking longer routes and being far away from each other. Hence, the distance travelled is very high. The time required by the robots increases due to the increased distance. As more robots start travelling away from the corridor with an increased value of the parameter, the speed increases from the high congestion low speeds to the free-flow speeds, almost equal to the maximum permissible value baring the mandatory speed drop due to start from a congested scenario. Hence, an intermediate value of α is desirable. α is the only important parameter of the algorithm that governs the performance and needs to be properly set.

Figure 12 shows the metrics with different values of α. All units are arbitrary and specific to the simulation tool. The results are averaged for 5 runs, and further, the curves are smoothened by using moving average. Travelling by using the shortest path, without using the routing mechanism, is similar to the system with very small α. The initial peak at a small value of α clearly shows the limitations of not routing. The simulations are stochastic, and therefore, the figures will have some deviation. Hence, the peak is not at 0 but at 1. A moderate value of α gives enough incentive for the robots to avoid congestions, and hence, a moderate value of α is desirable. The robots travel larger distances, however with greater speeds and take lesser total amount of time.

5 Conclusions

The paper presented a mechanism to move a large number of robots from their source to their goal using a deliberative probabilistic roadmap-based planning, an active re-planning, a reactive navigation system, a density conscious routing system and a deadlock avoidance system. The biggest challenge of the domain is to completely replicate the intelligence of the

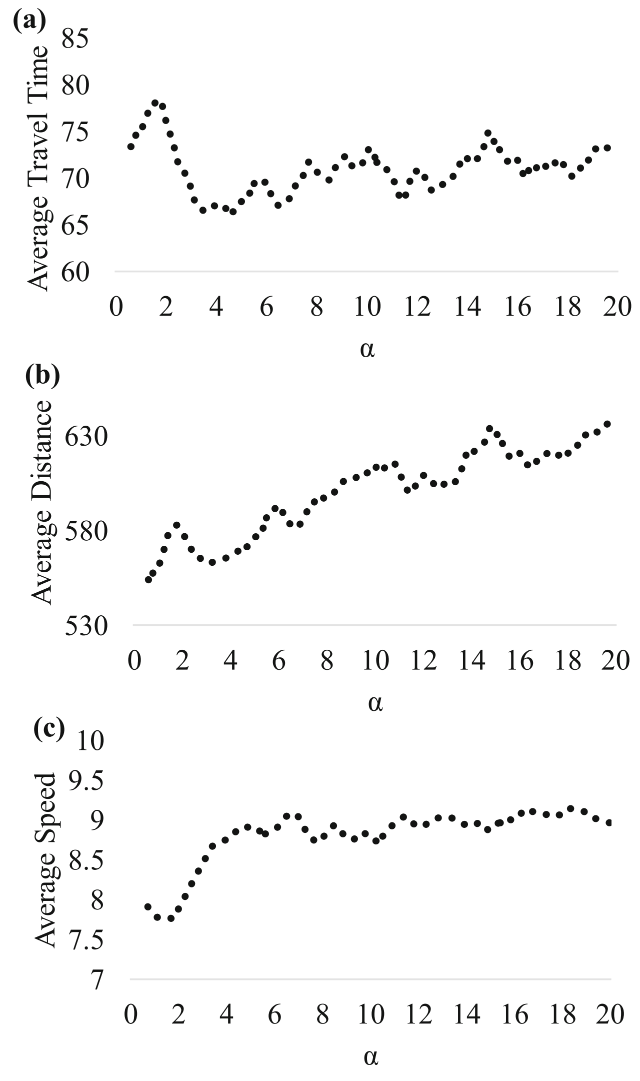


Fig. 12 Performance for changing values of penalty constant, α

humans in such crowded scenarios, who show natural traits of self-organization, cooperation and navigate as per mutual understanding. The results clearly display that the robots moved using trajectories that are very similar to humans. This is a very strong indicator that the human motion even in large densities of crowds is reactive in nature. In fact, the results are demonstrated using very basic rules of navigation, which further suggests that the human motion is not very complex and decisions are simple reactions to the precepts.

At the same time, the benefits of deliberation in navigation are clearly visible from the results. The navigation was much more efficient by using density-based routing than the navigation without density-based routing. The humans also use the known densities for all routing-based decisions. The densities on the routes near the current positions mainly affect decision-making. Even if the algorithm makes a route based on densities of faraway places, the re-routing will happen later and the routes will be corrected. At the time of working it appeared that the lack of anticipation will result in a severe loss in performance. Many times the humans can predict the intents of the people around based on the past experiences. On a stronger analysis, it appears that the humans do not really anticipate much, which explains taking a highway which after joining turns out to be congested; going by the shortest route which appears clear, not realizing everybody will take the same route; taking lift instead of stairs only to later realize that everybody coming out will do the same, etc. So even though the non-anticipatory system is not efficient, the traits of such non-efficiency also exist in the humans. Overall, this is a good framework to simulate the motion of a very large number of robots so as to effectively study the navigation intelligence of multi-robotic systems. The framework can be used to simulate a very large number of robots will modest computation costs.

Even though the paper claims the navigation model is closer to humans, the paper does not claim a complete imitation of the human motion. A deep analysis in a socialistic context requires a ground truth in the form of human navigation data. For this case specifically, the data have to be macroscopic in nature. No such data however exist that can be used for benchmarking. Even if such data are recorded, the problem is that the navigation is applied on the robots that have different kinematic and dynamic constraints, and form a different socialistic class in contrast to the humans. The human data cannot be transferred directly to the robots. Hence, the approach in this paper is intuitive in nature, that is, to observe good human navigation behaviours and to interpret the same in the robot's context. Further, the human motion can incorporate numerous factors like minimizing distance, minimizing energy, minimizing time or maximizing comfort. The travel may change depending upon the emotional status, whether the human is getting late, the preceding events, etc. Further, different humans have different styles of navigation and different preferences, so the crowd dynamics may change with the constituent of the crowd. Hence, completely mimicking humans may neither be advisable due to basic differences with humans nor possible due to unknown factors. The interpretation of human like is purely due to interpretation of the routing behaviour as observed in humans in a robotic context. In other words, the claim is that the robots are able to display a behaviour normally displayed by the humans, hence indicating some human intelligence traits.

The paper does enable routing of the robots based on the current densities. The same needs to be extended to anticipated densities without communication between the robots. The deadlock detection and avoidance schemes need to be improved so as to minimize false positives and reduce the time wasted in the detection. Further, the humans can organize themselves very well using rules such as keep on left/right, crowd entering travels before crowd leaving, under high density some routes may dynamically be used as one ways, etc. All these need to be implemented in the proposed approach to make the overall navigation even better.

References

1. Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementations. MIT Press, Cambridge
2. Tiwari R, Shukla A, Kala R (2013) Intelligent planning for mobile robotics. IGI Global Publishers, Hershey
3. Chand P, Carnegie DA (2012) A two-tiered global path planning strategy for limited memory mobile robots. Robot Auton Syst 60(3):309–321
4. Kala R, Shukla A, Tiwari R (2010) Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning. Artif Intel Rev 33(4):275–306
5. Kavraki LE, Svestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12(4):566–580
6. Bohlin R, Kavraki LE (2000) Path planning using lazy PRM. In: Proceedings of the IEEE international conference on robotics and automation, pp 521–528
7. Claes R, Holvoet T, Weyns D (2011) A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. IEEE Trans Intell Transp Syst 12(2):64–373
8. Kala R, Warwick K (2015) Congestion avoidance in city traffic. J Adv Transp 49(4):581–595
9. Spears WM, Spears DF, Hamann JC, Heil R (2004) Distributed, physics-based control of swarms of vehicles. Auton Robot 17(2–3):137–162
10. Pelechano N, Allbeck JM, Badler NI (2007) Controlling individual agents in high-density crowd simulation, In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp 99–108
11. Musse SR, Thalmann D (2001) Hierarchical model for real time simulation of virtual human crowds. IEEE Trans Virtual Comput Graph 7(2):152–164
12. Curtis S, Best AP, Manocha D (2016) Menge: a modular framework for simulating crowd movement. Collect Dyn 1(A1):1–40
13. Gu Q, Deng Z (2011) Context-aware motion diversification for crowd simulation. IEEE Comput Graph Appl 31(5):54–65
14. Kountouriotis V, Thomopoulos SCA, Papelis Y (2014) An agent-based crowd behaviour model for real time crowd behavior simulation. Pattern Recogn Lett 44:30–38
15. Han Y, Liu H, Moore P (2017) Extended route choice model based on available evacuation route set and its application in crowd evacuation simulation. Simul Model Pract Theory 75:1–16
16. Golas A, Narain R, Curtis S, Lin MC (2014) Hybrid long-range collision avoidance for crowd simulation. IEEE Trans Vis Comput Graph 20(7):1022–1034
17. Cowlagi RV, Tsiotras P (2012) Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. IEEE Trans Robot 28(2):379–395
18. Sgorbissa A, Zaccaria R (2012) Planning and obstacle avoidance in mobile robotics. Robot Auton Syst 60(4):628–638
19. Chang YC, Yamamoto Y (2009) Path planning of wheeled mobile robot with simultaneous free space locating capability. Intell Serv Robot 2(1):9–22
20. Yao Z, Gupta K (2011) Distributed roadmaps for robot navigation in sensor networks. IEEE Trans Robot 27(5):997–1004
21. Clark CM (2005) Probabilistic road map sampling strategies for multi-robot motion planning. Robot Auton Syst 53:244–264
22. Chai R, Su J (2013) Motion planning for multi-robot coordination. In: 13th IFAC symposium on large scale complex systems: theory and applications, Shanghai, China, pp 129–134
23. Alvarez-Sanchez JR, de la Paz Lopez F, Troncoso JMC, de Santos Sierra D (2010) Reactive navigation in real environments using partial center of area method. Robot Auton Syst 58(12):1231–1237
24. Sezer V, Gokasan M (2012) A novel obstacle avoidance algorithm: follow the gap method. Robot Auton Syst 60(9):1123–1134
25. Kim Y, Kwon SJ (2015) A heuristic obstacle avoidance algorithm using vanishing point and obstacle angle. Intell Serv Robot 8(3):175–183
26. Fiorini P, Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. Int J Robot Res 17(7):760–772
27. Snape J, van den Berg J, Guy SJ, Manocha D (2011) The hybrid reciprocal velocity obstacle. IEEE Trans Robot 27(4):696–706
28. Rashid AT, Ali AA, Frasca M, Fortuna L (2012) Multi-robot collision-free navigation based on reciprocal orientation. Robot Auton Syst 60(10):1221–1230
29. Karagoz CS, Bozma HI, Koditschek DE (2014) Coordinated navigation of multiple independent disk-shaped robots. IEEE Trans Robot 30(6):1289–1304
30. Zhong J, Cai W, Lees M, Luo L (2017) Automatic model construction for the behaviour of human crowds. Appl Soft Comput. https://doi.org/10.1016/j.asoc.2017.03.020
31. Patil S, van den Berg J, Curtis S, Lin MC, Manocha D (2011) Directing crowd simulations using navigation fields. IEEE Trans Vis Comput Graph 17(2):244–254
32. Hsu D, Sanchez-Ante G, Sun Z (2005) Hybrid PRM sampling with a cost-sensitive adaptive strategy. In: Proceedings of the 2005 IEEE international conference on robotics and automation, Barcelona, Spain, pp 3874–3880
33. Rodriguez S, Thomas S, Pearce R, Amato NM (2008) RESAMPL: a region-sensitive adaptive motion planner. In: Algorithmic Foundation of Robotics VII, Springer Tracts in Advanced Robotics. Springer, Berlin, pp 285–300
34. Morales M, Tapia L, Pearce R, Rodriguez S, Amat NM (2005) A machine learning approach for feature-sensitive motion planning. In: Algorithmic Foundations of Robotics VI, Springer Tracts in Advanced Robotics, Springer, Berlin, vol 17, pp 361–376
35. Kala R (2016) Homotopy conscious roadmap construction by fast sampling of narrow corridors. Appl Intell 45(4):1089–1102
36. Hsu D, Jiang T, Reif J, Sun Z (2003) The bridge test for sampling narrow passages with probabilistic roadmap planners. In: Proceedings of the 2003 IEEE international conference on robotics and automation, vol 3, pp 4420–4426
37. Amato NM, Bayazit OB, Dale LK, Jones C, Vallejo D (1998) Obprm: an obstacle-based prm for 3d workspaces. In: Agarwal P, Kavraki LE, Mason M (eds) Robotics: the algorithmic perspective. A.K. Peters, Natick, pp 155–168
38. La Valle SM, Kuffner JJ (2001) Randomized kinodynamic planning. Int J Robot Res 20(5):378–400
39. Kuffner JJ, LaValle SM (2000) RRT-connect: an efficient approach to single-query path planning. In: Proceedings of the IEEE international conference on robotics and automation, pp 995–1001
40. Russell S, Norvig P (2010) Artificial intelligence: a modern approach, 3rd edn. Pearson, Harlow
41. Helbing D, Molnar P (1995) Social force model for pedestrians dynamics. Phy Rev E 51(5):42–82
42. Martinez-Garcia E, Torres-Cordoba R (2012) Exponential fields formulation for WMR navigation. J Appl Bion Biomech 9(4):375–397