Robotics and Autonomous Systems 62 (2014) 474–486

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Guided Autowave Pulse Coupled Neural Network (GAPCNN) based real time path planning and an obstacle avoidance scheme for mobile robots

Usman Ahmed Syed *, Faraz Kunwar, Mazhar Iqbal

Probabilistic Robotics and Intelligent Systems in Motion (PRISM) Lab, Mechatronics Engineering Department, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad, Pakistan

HIGHLIGHTS

- Investigates the shortcomings of the MPCNN model proposed recently for mobile robots.
- Directional constraints on the autowave for informed search of configuration space.
- Dynamic thresholding of the underlying neural network to increase time efficiency.
- On ground implementation that verifies the efficacy of the proposed algorithm.

ARTICLE INFO

Article history:
Received 6 January 2013
Received in revised form
7 December 2013
Accepted 9 December 2013
Available online 27 December 2013

Keywords:
Path planning
Obstacle avoidance
GAPCNN
Neural networks

ABSTRACT

Real time path planning for mobile robots requires fast convergence to optimal paths. Most rapid collision free path planning algorithms do not guarantee the optimality of the path. In this paper we present a Guided Autowave Pulse Coupled Neural Network (GAPCNN) approach for mobile robot path planning. The proposed model is a novel approach that improves upon the recently presented Modified PCNN (MPCNN) by introducing directional autowave control and accelerated firing of neurons based on a dynamic thresholding technique. Simulation studies and experimental results in both static as well as dynamic environments confirm GAPCNN to be a robust and time efficient path planning scheme for finding optimal paths.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The development of an Intelligent Transportation System (ITS), which can drive autonomously to a desired location, requires solving such problems as Localization, Path Planning and Autonomous Obstacle/Collision Avoidance. One of the earliest and most well-known problems for such a system specifically for indoor domains, is the generation of collision free global path for the robot to move to a given point in a dynamic environment. Conventional path planning approaches employed for such motion planning can be categorized into four types including Visibility graphs [1,2], Cell Decomposition [3], Voronoi Diagrams [4] and the Potential Field methods [5,6]. Most of these algorithms suffer from time inefficiency in their computation and are not designed for use in real-time path planning. Also, Potential Field methods are known to suffer unwanted local minima in which the robot gets stuck in a U-shaped obstacle [7].

Sampling-based algorithms such as Probabilistic Road Maps [8,9] improve on the limitations of the previously mentioned algorithms but because of their iterative nature, they are not well suited to real-time implementations in highly cluttered environments. Techniques that use the Monte Carlo based growing data structures like Rapidly Growing Random Trees [10,11], can find paths in complex environments but they are proven not to approach optimality [12]. A recently proposed sampling based variant, RRT* [13], however, does approach optimality but requires infinite iterations. Search algorithms (e.g A* [14]) have also been employed for global free path searching. More recently genetic algorithms [15,16] and fuzzy approaches [17,18] have been proposed for collision free path planning.

Neural network approaches have also been employed for efficiently planning the shortest path. In this regard, the supervised as well as unsupervised learning techniques with neural networks have been employed to path planning and trajectory planning problems (e.g. [19–24]), both for manipulators and mobile robots.

* Corresponding author. Tel.: +92 3325743938.
E-mail addresses: s.usman87@hotmail.com, s.usman@ceme.nust.edu.pk (U.A. Syed), k.faraz@ceme.nust.edu.pk (F. Kunwar), drmazhar@ceme.nust.edu.pk (M. Iqbal).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.12.004

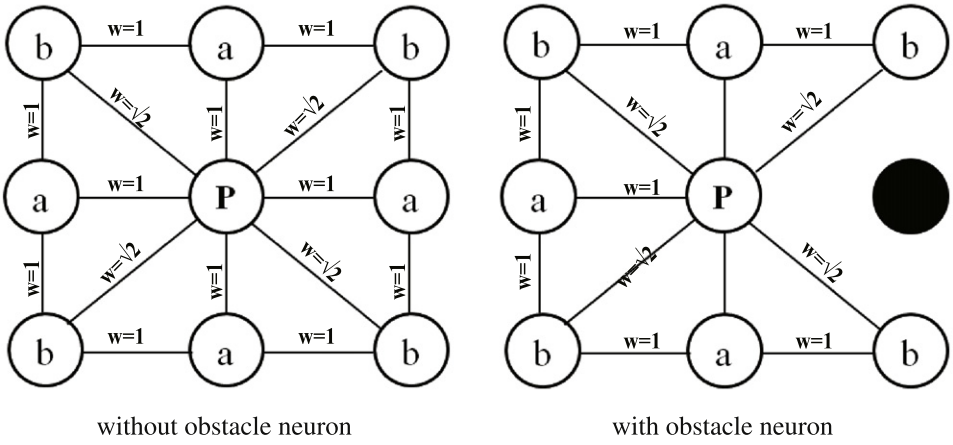


Fig. 1. Neighborhood connections in MPCNN architecture.

Glasius et al. [19] devised a method based on the Hopfield network for motion trajectory generation while avoiding obstacles but it suffers local minima. Li [22] has proposed a neural network based path planner by fusing an adaptive sensory model with a visual error correction model. However his method assumes an error free world, hence it cannot be used in real world sceneries. A much better approach which utilizes unsupervised learning paradigms, capable of handling dynamic environments has been proposed in [23] but it suffers in real world conditions since it is a computationally complicated solution. A reinforcement learning based multi layered path planning architecture has been proposed in [24] but it results in suboptimal trajectories especially in the learning phase. More recently a cellular neural network [25] and a reinforcement learning [26] based algorithm have also been presented, however these also suffer with similar optimality issues.

Another promising approach is the use of a Pulse Coupled Neural Network (PCNN) which is a biologically inspired algorithm capable of emulating the behavior of a cat's visual cortices [27]. After the earlier work on PCNN done by Johnson [28–30], the algorithm has been subject to extensive research in various fields such as image processing and pattern recognition [31]. By employing the autowave approach in the PCNN, Caulfield and Kinser [20] proposed a method to solve maze problems. Their approach finds the shortest path where computational complexity is related to the path length. However this approach needs a large number of neurons to find paths in large mazes thereby making it computationally expensive. Improving on the limitations of the aforementioned PCNN approach, a modified PCNN (MPCNN) model was recently introduced by Qu et al. [21] which has been proven to be more efficient than most planning algorithms. The modified model requires fewer neurons than the Caulfield and Kinser model, guaranteeing the shortest path and a solution that is independent of the complexity of the search space. However, since the model employs an unconstrained autowave, it searches the whole space irrespective of where the target is located, hence this unconstrained search leads to time inefficiency.

In order to facilitate an informed search, we present a novel path planning approach that employs a directionally constrained Guided Autowave Pulse Coupled Neural Network (GAPCNN) model. The model, which works on 2D space representation of 3D environments, uses the position of target neuron to focus a controlled search in its direction. Also, it employs a variable threshold unique to each neuron. These modifications enable the proposed scheme to significantly improve the optimal path query times, in both static and dynamic settings, as compared to the MPCNN.

Rest of the paper is organized in the following manner. Section 2 covers some related research on MPCNN based path planning. The proposed GAPCNN model is described in Section 3. Path planning algorithm using the model is given in Section 4. Section 5 deals with the comparison of MPCNN and GAPCNN in terms of their mathematical model. Performance comparison of GAPCNN with contemporary algorithms is presented in Section 6 while Section 7 presents a brief overview of the implementation details. In Section 8, experimental results are discussed. Finally conclusions are given in Section 9.

2. MPCNN based path planning

Here we briefly review some fundamental concepts of MPCNN algorithm for mobile robot path planning. Path planning using MPCNN utilizes a topologically organized, latterly connected 2D lattice of neurons over the entire configuration space of the environment as shown in Fig. 2. Each MPCNN neuron receives two inputs, one from neighboring neurons while the other corresponds to the configuration space input. All the neurons are designated as either obstacle or free space, both robot and target are treated as free space. Each neuron in the space is connected only to its immediate neighbors (Fig. 1(a)). Obstacle neurons are not connected to their neighbors as shown in Fig. 1(b).

The neighbors of a particular neuron are said to be in its straight connected set N^s (a-type neurons, as shown in Fig. 1) if their Euclidean distances from the neuron are 1. Likewise the neighbors are said to be in the slantwise connected set N^d (b-type neurons, as shown in Fig. 1) of the neuron if their Euclidean distances are $\sqrt{2}$. The neurons need not be actually spaced a unit distance (or $\sqrt{2}$ in slantwise direction) apart from each other. The neurons can be placed at greater distances for less cluttered environments, provided the adherence to straight and slantwise connection rules is ensured.

A neuron i is connected to any of its immediate neighbor j through a weight w_{ij} as given below:

$$w_{ij} = \begin{cases} 1 & \text{if } j \in N_s - \text{a-type neurons} \\ \sqrt{2} & \text{if } j \in N_d - \text{b-type neurons.} \end{cases} \quad (1)$$

Output of a neuron i at time t is represented by $Y_i(t)$. Each neuron is fired only once in its life time.

$$Y_i(t) = \begin{cases} 0 & \text{for } T - \varepsilon \le t < T \\ 1 & \text{for } t = T \\ 0 & \text{for } T < t \le T + \varepsilon \end{cases} \quad (2)$$

where ε could be any amount of time and $T = t^i_{fire}$ is the time at which neuron i fires.

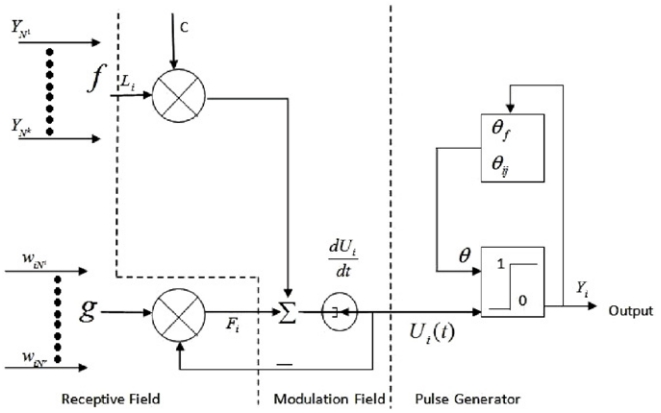


Fig. 2. The MPCNN model.

A neuron i fires only if some neuron j in its neighborhood N_i fires and this sets neuron i as child neuron to fire at some later time. The neuron j is said to be its temporary parent N_i^{P*} and its firing time is denoted as $t^{N_i^{P*}}$. If another neuron in the same neighborhood fires and if it can make neuron i fire earlier, then it will become the new temporary parent of i. When a neuron i fires at some time t_{fire}^i it records its current temporary parent as its parent $t^{N_i^P}$.

Firing of these neurons is governed by their energy $U(t)$ also known as internal activity. A neuron is activated by its parent neuron and its internal activity before and at the time of activation by its parent is zero. At time $t > t^{N_i^P}$, its internal activity is given as follows:

$$\begin{cases} \dfrac{dU_i(t)}{dt} = F_i + CL_i & \text{for } t > t^{N_i^P} \\ U_i(t) = 0 & \text{for } t \le t^{N_i^P} \end{cases} \tag{3}$$

where C is a constant $F_i(t)$ and $L_i(t)$ are the linking and feeding fields given by:

$$L_i(t) = f(Y_{N^1}, \ldots, Y_{N^k}, t) = \begin{cases} 0 & \text{if } t < t^{N_i^{P*}} \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Linking input is the actual input from the environment map whereas the feeding field is due to the connections with the neighboring neurons.

$$F_i(t) = -g(w_{ir^1}, \ldots, w_{ir^k}, t)U_i(t). \tag{5}$$

Here g is a function of time t and connection weights among the k neighbors and is given by:

$$g(w_{ir^1}, \ldots, w_{ir^k}, t) = \begin{cases} 0 & \text{if } t < t^{N_i^{P*}} \\ \mu(w_{ij}) & \text{if } t > t^{N_i^{P*}} \end{cases} \tag{6}$$

where $\mu(w_{ij})$ is given by:

$$\mu(w_{ij}) = \frac{B}{w_{ij}} = \begin{cases} B & \text{if } j \in N_s \\ \dfrac{B}{\sqrt{2}} & \text{if } j \in N_d. \end{cases} \tag{7}$$

Here B is a positive constant. Depending upon weather the child is a straight neighbor or a diagonal neighbor of parent neuron, the internal activity $U_i(t)$ of the neuron increases in accordance with Eq. (3). Energy of the neuron is continuously compared with the threshold level. The threshold for ith neuron can be written as:

$$\theta_i(t) = \begin{cases} \theta_{init} & \text{for } t < t^{N_i^{P*}} \\ \theta_{ij} & \text{for } t^{N_i^{P*}} \le t < t_{fire}^i \\ \theta_f & \text{for } t > t_{fire}^i \end{cases} \tag{8}$$

where θ_{init} and θ_f are constants. θ_f is set to a very large value to ensure that the neuron may not fire again and θ_{ij} is given as:

$$\theta_{ij} = \begin{cases} \theta_s & \text{if } j \in N_s \\ \theta_d & \text{if } j \in N_d \end{cases} \tag{9}$$

where j is the temporary parent of neuron i. θ_s and θ_d are positive constants. These constants determine the threshold level for the straight and diagonal neurons respectively. A neuron fires when its internal activity exceeds a threshold level: θ_s in case of a straight connected neuron and θ_d in case of a diagonal connected neuron. Output of a MPCNN neuron is given by the following relation:

$$Y_i(t) = Step(U_i(t) - \theta_i(t)) = \begin{cases} 1 & \text{if } U_i(t) \ge \theta_i \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Aforementioned architecture of a MPCNN neuron can be visualized as a model in Fig. 2.

To find the shortest path using MPCNN, search proceeds through the use of a circular autowave. The autowave propagates through coupling of the neighboring neurons. When the internal activity of a particular neuron exceeds the threshold level as given by Eq. (10), it fires. Initially the target neuron's internal activity is set greater than the threshold level which is a necessary condition to initiate the firing process. The firing of a neuron is followed by coupling process in which neighboring neurons are bonded by a parent–child relationship. Subsequently the internal activity of the neighboring neurons increases with time. This firing pattern propagates outwards in the form of a wave until the robot neuron is reached. In this process, internal activity of the obstacle neurons is kept zero, hence avoiding their firing. Path is traced backwards from robot to target through the parental sequence of neurons.

An analysis of the MPCNN algorithm reveals that it suffers in two important aspects. Firstly it utilizes an uninformed search behavior. To find the shortest path it propagates a wave in all directions, not using any of the available information to perform an intelligent search. Secondly MPCNN assigns the same threshold level to all the neurons in the provided map. As we demonstrate in this paper, gradual decrease in the threshold in the direction of the possible solution can be utilized to find the shortest path with better time efficiency. We term the new model as Guided Autowave PCNN (GAPCNN). GAPCNN utilizes informed search and threshold constraints by employing mathematical expressions of least computational complexities; thereby leading to a computationally efficient solution to the presented problem. Moreover, firing rate based wavefront expansion ensures its completeness. GAPCNN outperforms previous methods with the same path planning results, but with a much greater time efficiency.

3. GAPCNN model

The GAPCNN model for a single neuron i is shown in Fig. 3. The model differs from the MPCNN model in two ways. First the threshold function of the proposed model is a function of time t and distance λ from the target while the MPCNN threshold is a function of time only. The modified threshold has distinct values over the whole space and it decreases towards the target neuron. The threshold for ith neuron can be written as:

$$\theta_i(t, \lambda) = \begin{cases} \theta_{init} & \text{for } t < t^{N_i^{P*}} \\ \theta_{ij} - \lambda_i & \text{for } t^{N_i^{P*}} \le t < t_{fire}^i \\ \theta_f & \text{for } t > t_{fire}^i. \end{cases} \tag{11}$$

λ_i is the normalized distance of the ith neuron from the target neuron given by:

$$\lambda_i = A\left(1 - \frac{D_{target}}{D_{max}}\right). \tag{12}$$

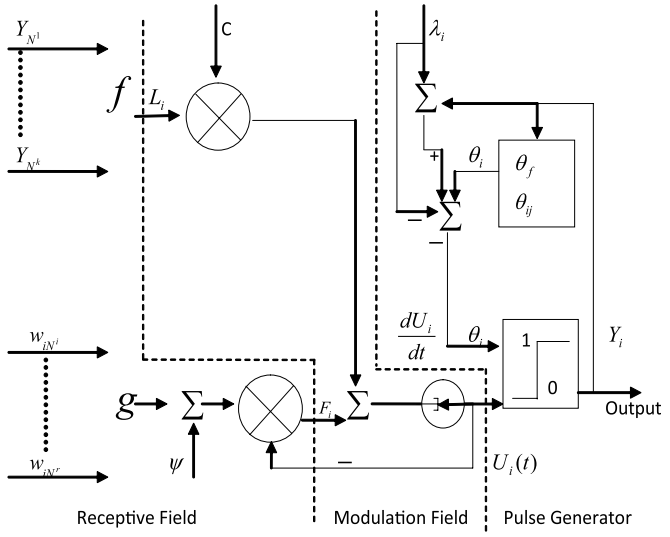


Fig. 3. The GAPCNN model.

Here D_{target} is the Euclidean distance from the target neuron, D_{max} is taken as the distance between the two farthest neurons in the network, and A is a constant.

Second major difference of the proposed model from the MPCNN model is the actual directional constraining of the autowave. This is achieved by employing an angle modification in the original differential equation of the internal activity of the neuron. Modifying Eq. (3), the proposed internal activity $U_i(t)$ of GAPCNN neuron is given by:

$$\begin{cases} \dfrac{dU_i(t)}{dt} = F_i + CL_i \quad \text{for } t \geq t^{N_i^P} \\ U(t^{N_i^P}) = 0. \end{cases} \tag{13}$$

This is achieved by modifying the feeding field of the model as given by the following expression:

$$F_i(t) = -g(w_{iN^1}, \ldots, w_{iN^k}, \psi, t)U_i(t). \tag{14}$$

Here g is a function of time t, connection weights among the k neighbors and the directional constraint ψ and is given by:

$$g(w_{iN^1}, \ldots, w_{iN^k}, \psi, t) = \begin{cases} 0 & \text{if } t < t^{N_i^{P*}} \\ \mu(w_{ij}) + \psi_i & \text{if } t > t^{N_i^{P*}}. \end{cases} \tag{15}$$

The directional constraint ψ_i is given by:

$$\psi_i = K(\alpha - \beta_i) \tag{16}$$

where α is the principle angle of the robot from the target which is calculated by assuming a coordinate system with the origin at the target. β_i is the angle of the ith neuron from the target. The angle calculations are illustrated in Fig. 4, the orientation of this axis is predefined and all angle computations shall be carried out with reference to this orientation. K is given as:

$$K = p\delta. \tag{17}$$

Here p is the proportionality constant, δ is the difference in the firing rate of neurons at time step t and $t + 1$. A decrease in firing rate indicates obstruction in front of wavefront, which would consequently decrease K thereby opening the wavefront. Such a situation occurs in the presence of a local minima. In this case the wavefront is gradually opened to evade the local minima. If there does not exists a path from goal to robot neuron then in such a case based upon the values of δ, the wavefront is opened until GAPCNN approaches MPCNN. In the end δ would diminish to zero thereby indicating that the wavefront cannot continue forward and

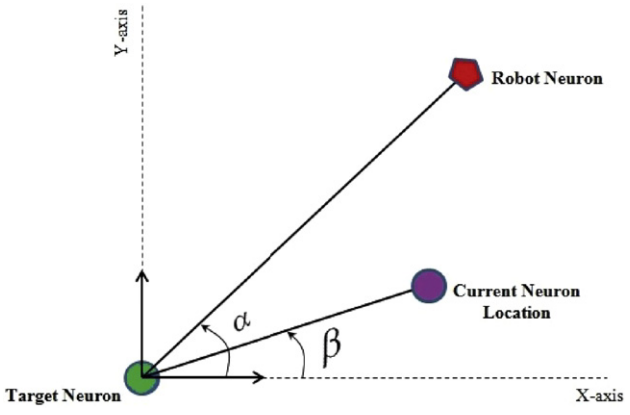


Fig. 4. Illustration of the direction constraint.

there does not exists a route from target to robot. This ensures the completeness of the algorithm i.e. the algorithm reports a path if present and reports that no path exists in case of failure.

The output $Y_i(t)$ of a neuron is given by:

$$Y_i(t) = Step(U_i(t) - \theta_i(t, \lambda)) = \begin{cases} 1 & \text{if } U_i(t) \geq \theta_i \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

4. Algorithm for path planning

Path is planned using the guided autowave starting from the target neuron which fires in the first cycle. From the target neuron there will be a sequence of children of the subsequent neurons in the path which ends at the robot. The directional parameter ensures that the neurons which have angles more closely aligned with the fundamental angle of the robot from the target, will fire earlier than the less aligned ones.

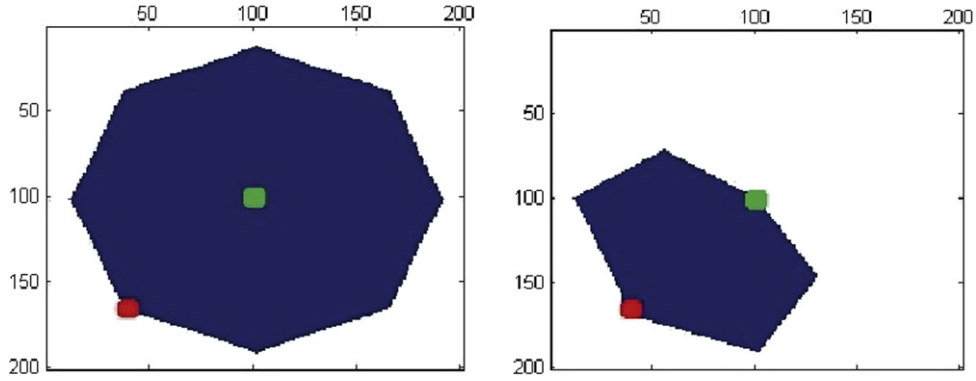
The computation steps are as follows:

(1) *Network Initialization*: In this step the various parameters of the system are set. For all neurons, we set $\theta_i(0) = \theta_{init}$ and set $U_i(0) = 0$ for all $i \neq target$, $U_i(0) = \theta_{init}$ for $i = target$,
(2) *Network Iterations*: For each neuron i
  (a) Calculate $U_i(t)$ according to (13)
  (b) Calculate $Y_i(t)$ according to (18)
  (c) If $Y_i(t) = 1$:
    (i) Set $\theta_i(t) = \theta_f$,
    (ii) Set N_i^{P*} as N_i^P,
    (iii) Calculate $\psi(i)$ according to (16),
    (iv) For all $j \in N_i$, if N_j^{P*} exists then set minimum of $\psi(i)$ as N_j^{P*}, reset initial condition $U_j(t^{N_j^{P*}}) = 0$ upon temporary neighbor change. If N_j^{P*} does not exist set i as N_j^{P*},
    (v) Calculate λ_j for all $j \in N_i$ according to (12) and θ_j according to (11).
  (d) Calculate δ, if δ decreases determine corresponding K.
  (e) If $Y_{robot} = 1$ stop the execution and return path.

The path points are retrieved by backtracking through the parent of each neuron starting from the robot.

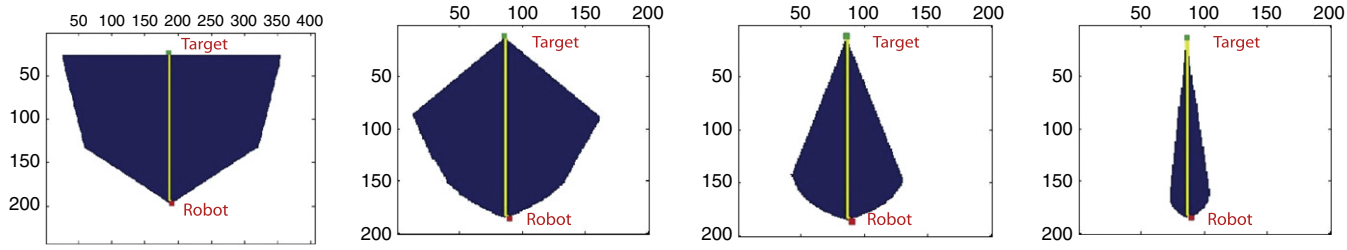
5. MPCNN vs. GAPCNN

GAPCNN is designed to avoid unnecessary propagation of the autowave by imparting a directional behavior and also to boost up the directed neuron's energy in order to accelerate neuron firing. GAPCNN inherits basic methodology from MPCNN, hence it is dependent upon various parameters introduced in MPCNN model. Optimal values of the parameters B, C, θ_s, θ_d and θ_f can be



(a) MPCNN. (b) GAPCNN.

Fig. 5. Search space.



(a) $K = 0.2$. (b) $K = 0.75$. (c) $K = 1$. (d) $K = 5$.

Fig. 6. Closing of autowave face with different values of K.

directly inferred from [21]. Acceptable range for these parameters are as follows:

$$0 < \theta_s < \frac{C}{B}, \qquad 0 < \theta_d < \frac{\sqrt{2}C}{B} \quad \text{and} \quad \theta_f > \frac{\sqrt{2}C}{B}.$$

Here the constant B and C can arbitrarily take any value provided aforementioned conditions are obeyed. Variation of the constants leads to change in the internal propagation time of the auto-wave. Here propagation time decreases as C increases while decrease in B lowers the propagation time but comparatively to a lesser extent. More detailed discussion about these parameters can be found in [21]. It is to be noted that these parameters are independent of the environment and same parameters are used for all the experiments reported in this research. Parameters A and K are introduced in the GAPCNN model and are discussed in detail as follows:

5.1. Wave propagation control

The search space of the original MPCNN and the GAPCNN is depicted in Fig. 5. In all the subsequent figures the blue region represents the search space of the algorithms, green shows the target neuron, red is the robot neuron while yellow represents the path planned by the scheme. The MPCNN wave propagates equally in all directions regardless of the location of the robot while the GAPCNN wavefront only propagates in a small region around the target reducing the number of neurons that have fired (blue area). Two parameters K and A are introduced in the GAPCNN model. K controls the opening and closing of the face of guided autowave. Fig. 6 demonstrates the influence of parameter K on the wave behavior. The face of the autowave opens with lower values of K and closes as K increases.

The magnitude of K is controlled based on the difference in the firing rate of neurons at each time step i.e. upon a decrease in the firing rate in successive iterations, the value of K increases, causing the wavefront to expand. Thus, whenever the directed wave gets stuck against an obstacle, the wavefront expands, allowing the autowave to go around the obstacle. This guarantees superior efficiency of GAPCNN over MPCNN even in scenarios where no relatively straight collision free path exists. Also when the rate of firing diminishes to zero, this will indicate that there is no route between the target and the robot thereby ensuring completeness of the approach.

Table 1
Execution time vs. parameter A.

Value of parameter A	Execution time (s)
0.01	1.0
0.5	0.85
1.0	0.71
3.0	0.59
6.0	0.28

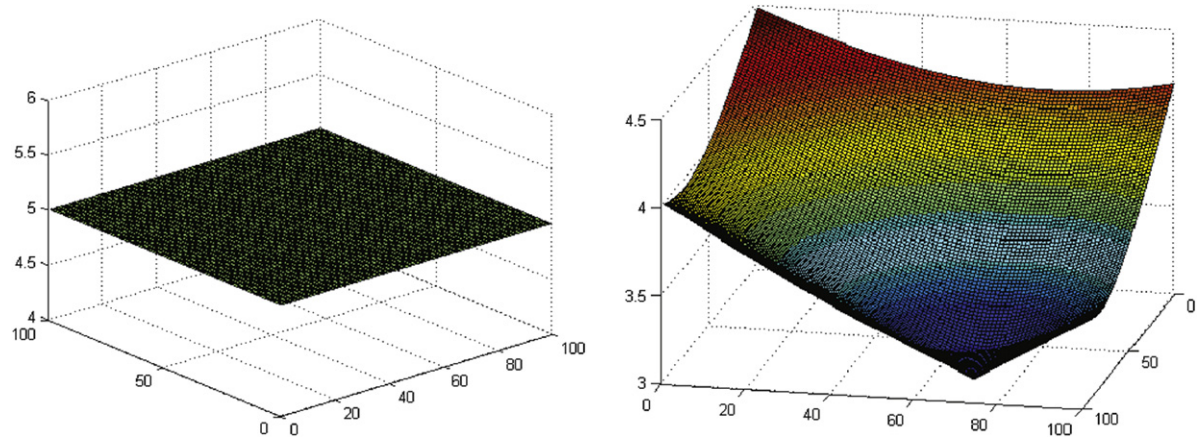
5.2. Dynamic threshold control

In GAPCNN each neuron of the grid possesses a unique threshold in accordance to its location whereas in MPCNN each neuron was associated with the same threshold level. It can be seen that while the MPCNN (Fig. 7(a)) has a constant threshold for all the neurons, GAPCNN threshold is dynamic and it significantly decreases as we approach the neurons in the vicinity of the robot neuron (Fig. 7(b)).

Dynamic thresholding improves the computational efficiency by allowing the neurons to fire earlier as compared to MPCNN. On the other hand it foils the possibility of false firing by providing a suitable threshold level to all the neurons. In the proposed model parameter A impacts the propagation speed of the autowave. Larger values in A decrease the wave propagation time as given in Table 1.

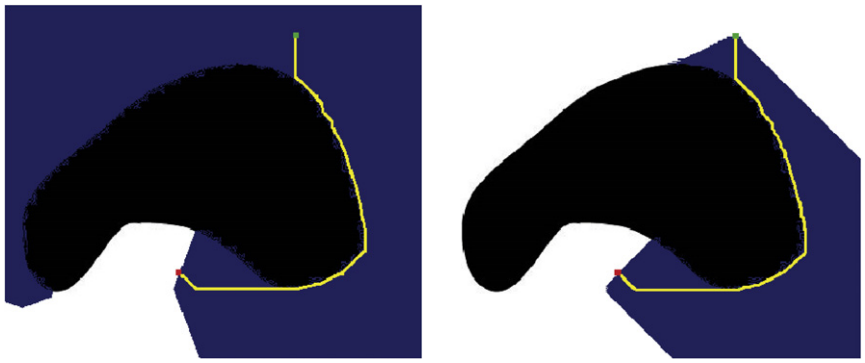
6. Performance comparison

To validate the effectiveness of the proposed algorithm, GAPCNN is compared with MPCNN. Additionally performance



(a) MPCNN. (b) GAPCNN.

Fig. 7. Threshold for grid of MPCNN and GAPCNN.



(a) MPCNN. (b) GAPCNN.

Fig. 8. Path planning in the presence of a local minima.

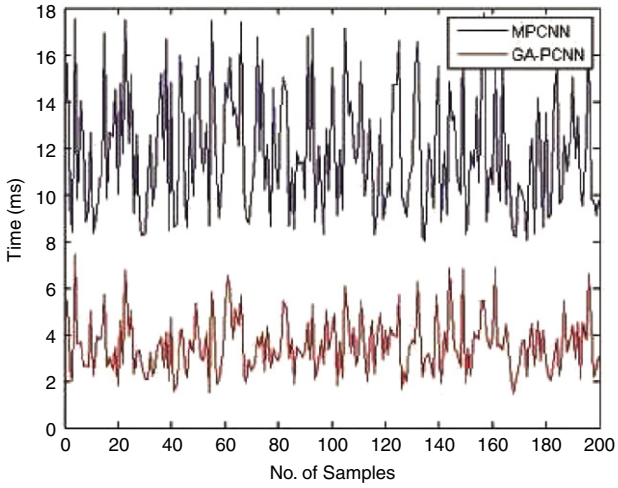


Fig. 9. Query times of MPCNN and GAPCNN.

comparison with A* search algorithm under identical conditions has also been conducted and presented in this section. Comparison results show that the proposed research yields much better results as compared to A* search.

6.1. Comparison with MPCNN

In order to gauge the performance improvements of GAPCNN in comparison with MPCNN, path query was performed in various situations. For the case of local minima (Fig. 8(a) and (b)), the proposed scheme successfully finds the path while still achieving a faster execution than the MPCNN. Note that the search space is almost halved. Graph shown in Fig. 9 presents the query times of the two algorithms for 200 test environments with randomly generated obstacle placements. The mean execution time of MPCNN for the results given in Fig. 9 is 11.82 ms whereas that of GAPCNN is 3.57 ms. The mean time improvement here is 8.25 ms which corresponds to a 70% time improvement. From Fig. 9 it is clear that in all of the path queries, GAPCNN performs significantly better than MPCNN in terms of time taken to determine the optimal path.

6.2. Comparison with A* search

A* [32] is a search algorithm which has been extensively employed to the problems of path planning and graph traversals. It determines the path with the smallest cost (represented by $f(x)$) in a graph or a network based upon the length of path already covered (given by $g(x)$)and a admissible heuristics ($h(x)$) representing an approximation of the path to be covered. Cost of each path is given as follows:

$$f(x) = g(x) + wh(x)$$

where w represents the weight function. A* works in as similar manner as that of GAPCNN i.e. first exploration is carried out based upon a cost function and then shortest path is extracted through parent–child relations. Comparison of the proposed algorithm with A* search has shown that GAPCNN is more efficient than A* in terms of time for the simulated scenarios. Fig. 10 shows a

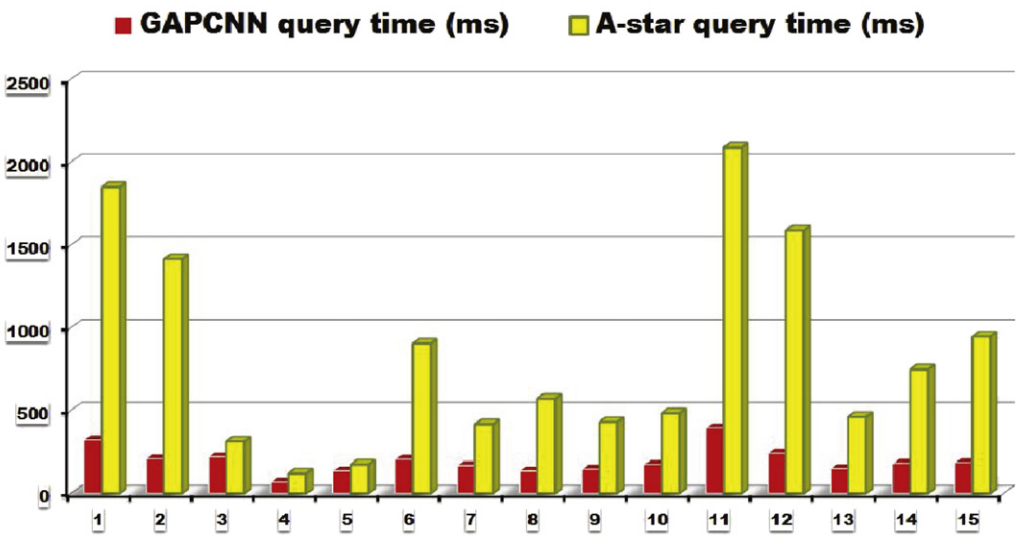


Fig. 10. Time comparison of GAPCNN with A* search.

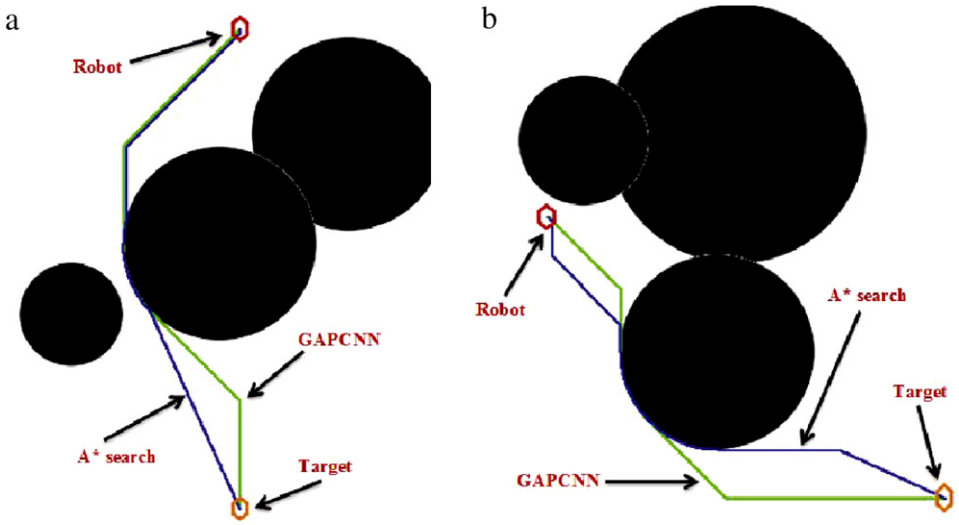


Fig. 11. Path comparison of GAPCNN with A* search.

comparison of the path query time for GAPCNN (shown in red) and A* (shown in yellow) which demonstrates that GAPCNN is much more efficient in terms of time as compared to A*. In all the cases cumulative cost of the resulting paths was equal, however it was observed that in most of the cases the paths calculated by GAPCNN were different from that of the path determined by A* as shown in Fig. 11(a) and (b). Implementation of A* was taken from [33].

7. System overview

The holistic overview of the GAPCNN implementation is given in Fig. 12. The implementation can be considered as a unison of three modules namely vision module, path planning module and lastly robot control module to follow the path provided by the path planner.

7.1. Vision module

This module deals with the objects location and pose identification, as shown in Fig. 12. Microsoft Kinect is used as the imaging source. This module takes as an input a RGB image and based upon this data identifies robot configuration (x_r, y_r, θ_r), static obstacle location (x_i^s, y_i^s) and dynamic obstacle locations (x_i^d, y_i^d). The obstacle locations are utilized in developing workspace of the environment. Since GAPCNN takes robot as a point robot and determines the optimal path, to avoid collisions with the obstacles it is necessary to increase the obstacles in proportion with the robot size. Hence we need to develop the configuration space of the robot.

7.2. Configuration space

The space of all possible configurations of a robot is called the configuration space [34]. Consider a robot R navigating in a 2-D Euclidean space, where the set of all possible configurations of the robot are represented by the set $Q = \{q_1, q_2, \ldots, q_n\}$ and the set of obstacles is represented by $O = \{O_1, O_2, \ldots, O_n\}$. Then the configuration space can be modeled as a continuous mapping represented by $\tau : [0, 1] \rightarrow Q$, where $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$. The path planning problem is to find a path in the configuration space such that no configuration of the robot collides with the obstacles.

Since the Robot can be of any arbitrary shape in the workspace, the profile of the robot needs also to be considered in the configuration space. This is done by taking the Minkowski Sum [35] of the profile of the robot R with every obstacle O_i.

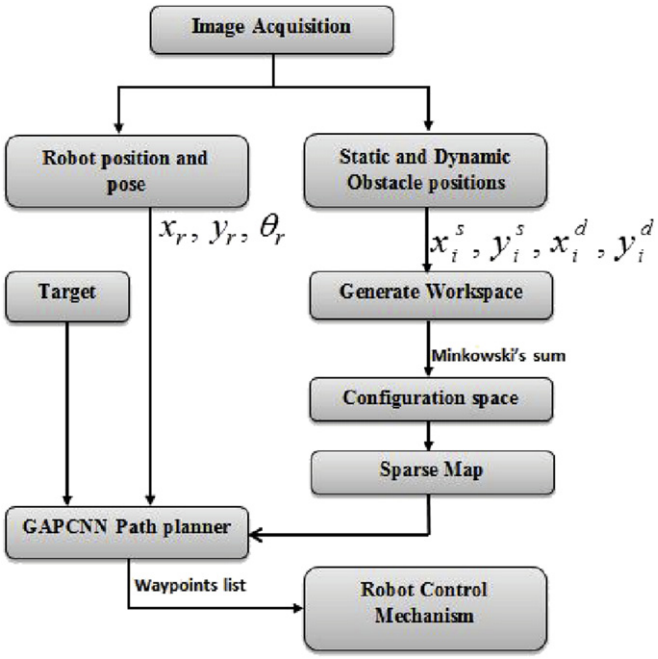


Fig. 12. Overview of the complete system.

Since the shape of the robot has been accounted for, the robot can now be considered as a point robot in the configuration space. In order to reduce the computational load, here the robot is assumed to have a circular shape. This removes the need to incorporate different robot orientations in the Minkowski sum. However this results in a minor degradation in the accuracy of the proposed system which is acceptable for the purpose of navigation in large indoor environments.

7.3. Sparse environment map

To decrease the computational load as well as to increase the path dynamics, the sparse representation of the configuration space is developed. Sparsing enables the path planner to result in a path which makes easier for the robot dynamics to follow the path. In our case we have split our configuration space to kernels of 3×3 size and examined all the kernels. If there exists an obstacle cell in the kernel then the corresponding cell in the sparse map shall be an obstacle cell otherwise its a free cell.

7.4. GAPCNN path planner and path following control

Using the robot location, the target location and the sparsed map, the GAPCNN is used to plan the path. GAPCNN requires a complete knowledge of the environment. The vision modules provides a configuration map of the environment, along with the robot location to the path planner and it plans a path with a predefined target location (in case of static targets). The path planner provides a list of the way-points to be followed by the robot.

Robot control mechanism module enables the robot to follow the way-points returned by the path planner. Based upon the current robot configuration and next location to be visited by the robot, the drive parameters i.e. robot speed and turnrate are determined. The robot is then transmitted the new speed and turnrate using Player/Stage [36]. Until the robot reaches the next way-point, the robot configuration is continuously determined to generate new control parameters. This process is repeated until the robot reaches the target location.

8. Results

Extensive testing of the algorithm has demonstrated its suitability for path planning in various real world scenarios. Here we present some of the results showing to prove the effectiveness of the proposed modifications in the light of obstacle avoidance, we shall present here the path planning using GAPCNN in both static as well as dynamic environments.

8.0.1. Dynamic environments

To ascertain the performance of the proposed method in dynamic world environments, simulations were carried out using Player/Stage. In Figs. 14–17 the bottom figures show the path planned for the robot by GAPCNN at the present time step given the current obstacle proximity and is drawn in green color. Black shows the obstacles whereas the robot is drawn in red. The corresponding figures at the top present the trajectories actually followed by the dynamic obstacles and the robot in the Player/Stage simulation until that time step. This figure includes replanned robot trajectory computed by GAPCNN to avoid the dynamic obstacle present in the vicinity of the robot. In Player/Stage representation there are two dynamic obstacles, one with a blue trajectory (obstacle 'A') and the other with a green trajectory (obstacle 'B') while the red trajectory shows the movement of the robot as determined by the GAPCNN-based intelligent path planning scheme. Table 2 shows velocities of robots and obstacles used for various experiments.

8.1. Simulation results

To verify the performance of aforementioned modifications simulation results were carried out. For dynamic environments, Player/Stage is used to verify the results.

8.1.1. Static environments

2D SLAM maps generated in different real world environments were used to test the performance of proposed algorithm. Fig. 13(a) and (b) present the paths planned in two different environments by GAPCNN. It can be observed in these figures that GAPCNN does not search the complete environment but follows a directed search pattern.

Fig. 14 presents a test case in which obstacle 'A' is on a head-on collision course with the path planning robot. A straight path exists from robot to target but its obstructed by this obstacle. Fig. 14(a) shows initial path planned to avoid head on collision with obstacle 'A'. Curvilinear path taken by robot 'R' in Fig. 14(b) and (c) shows the avoidance behavior of the proposed method. Next the robot is intercepted by obstacle 'B' and it evades collision by rerouting the path to the left as shown in Fig. 14(d). Fig. 14(e) shows the trajectories of robot and the obstacles.

Fig. 15 simulates a similar situation with an additional static obstacle. Fig. 15(a) and (b) show avoidance of possible collision with obstacle 'A'. Fig. 15(c) presents how collision with static obstacle is avoided. In Fig. 15(d), the robot determines that the optimal path is in front of obstacle 'B', and by moving in a curvilinear pattern it avoids obstacle 'B'.

Fig. 16 shows that the GAPCNN works perfectly and is independent of the shape of the obstacles. In this case obstacles are modeled as circular instead of square shaped. The overall trajectory in Fig. 16(e) shows that there is a significant difference in the path taken by robot 'R' as compared to previous case. The path is clearly modified by the GAPCNN to avoid the circular shaped obstacles.

Fig. 17 shows another generally encounter path planning problem in which the target is dynamic. Here the target is moving

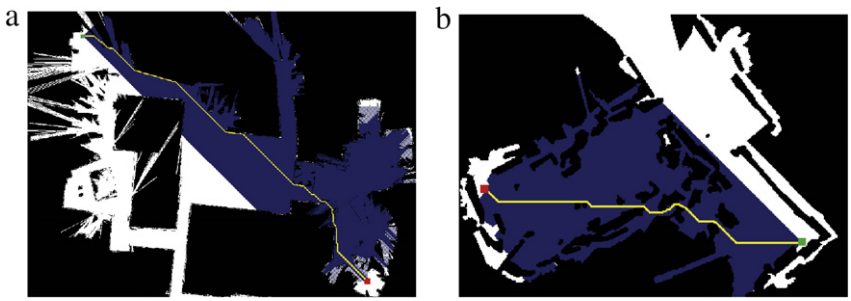


Fig. 13. Path planning in a 2D SLAM environment.

Table 2
Dynamic test cases.

Serial no.	Complexity	Obstacle A velocity (m/s)	Obstacle B velocity (m/s)	GAPCNN robot (m/s)
1	Two dynamic obstacles, obstacles modeled as squared shape	4.5	5	6.5
2	Two dynamic and one static obstacle, obstacles modeled as squared shape	4.5	2	7
3	Two dynamic obstacles, obstacles modeled as circular shape	3.5	1.5	6
4	Sinusoidal moving target intelligent pursuer	–	4	8

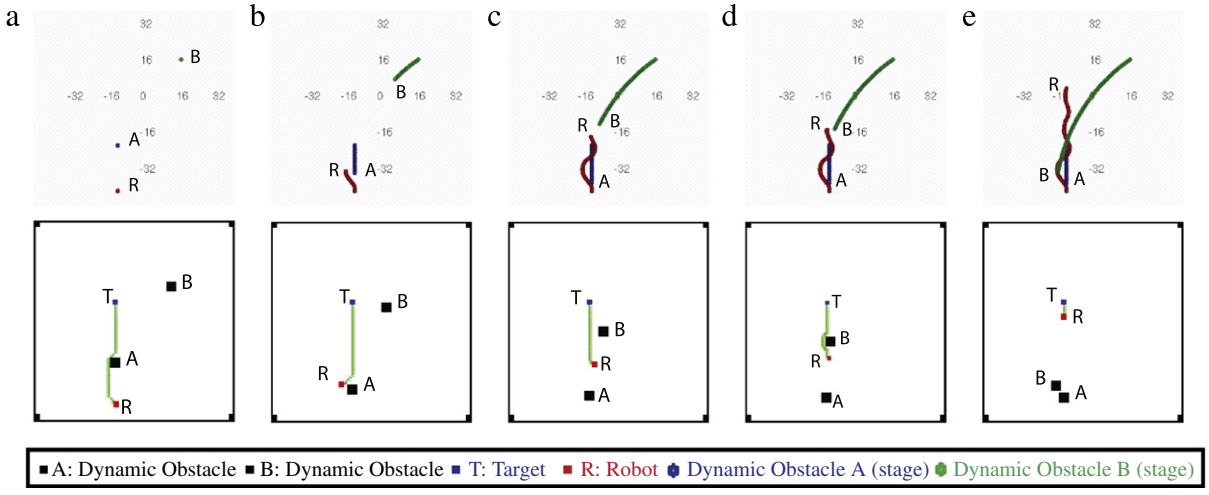


Fig. 14. Dynamic environment with two straight moving obstacles.

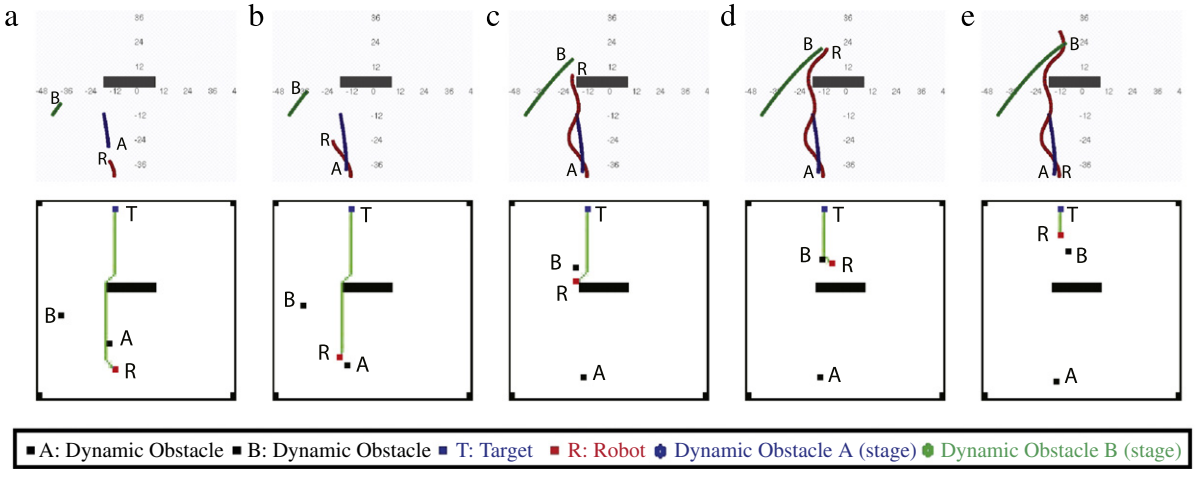


Fig. 15. Dynamic environment with two straight moving and one static obstacles.

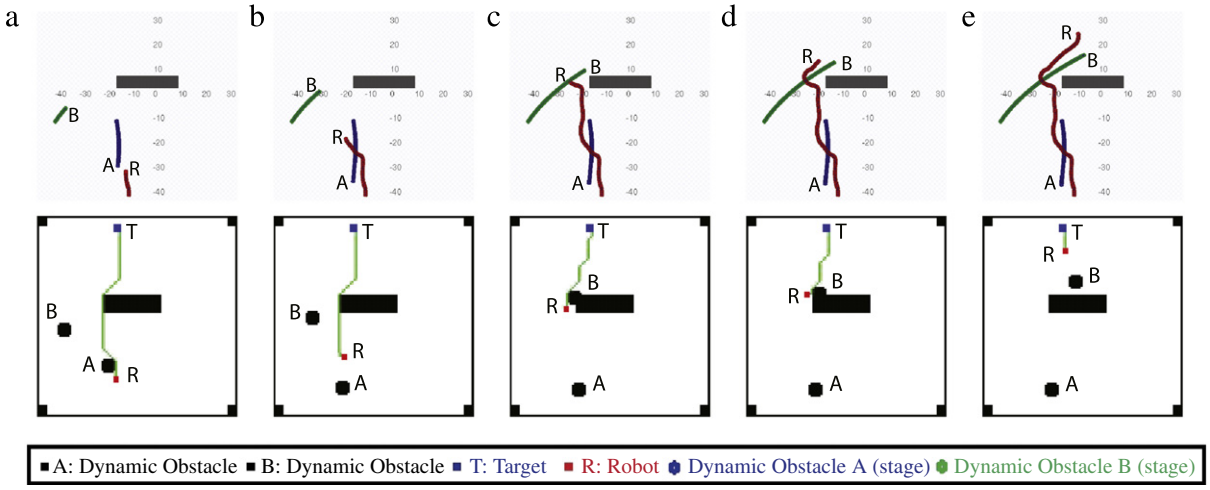


Fig. 16. Path planning with circularly modeled dynamic obstacles.

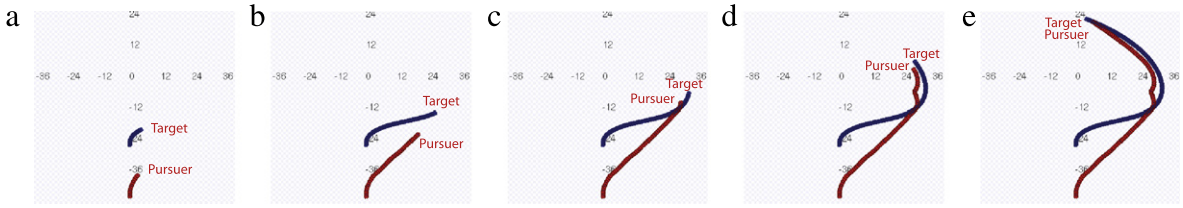


Fig. 17. Pursuit evasion with target moving in a sinusoidal manner.

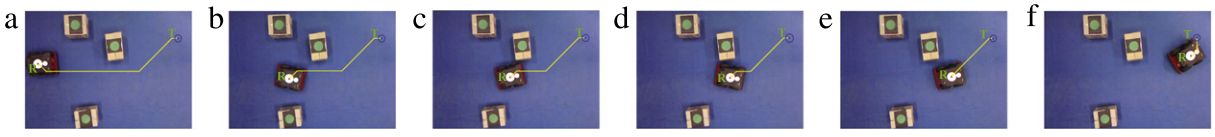


Fig. 18. Experimental results of GAPCNN path planning in static environment.

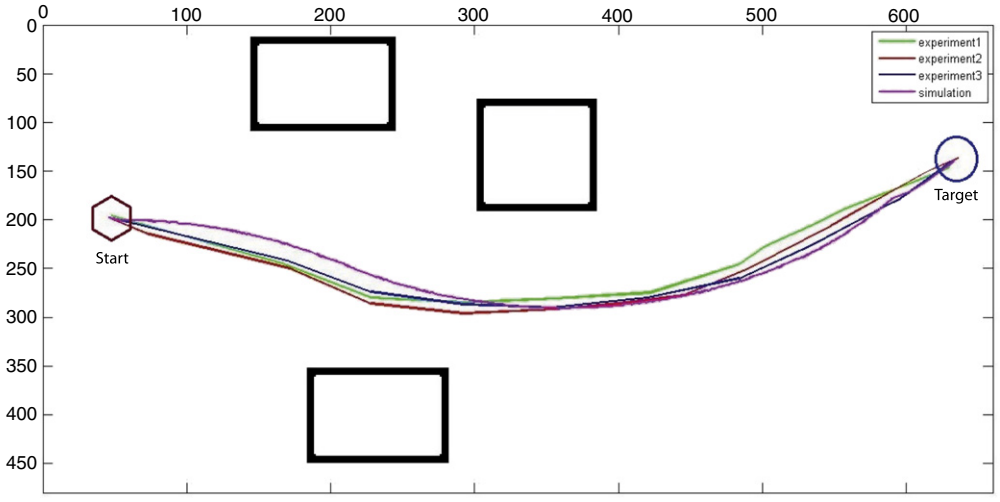


Fig. 19. Experimental comparison of paths generated by GAPCNN.

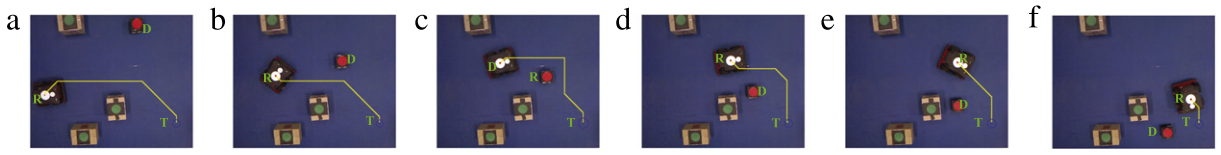


Fig. 20. Experimental results of GAPCNN path planning in dynamic environment.

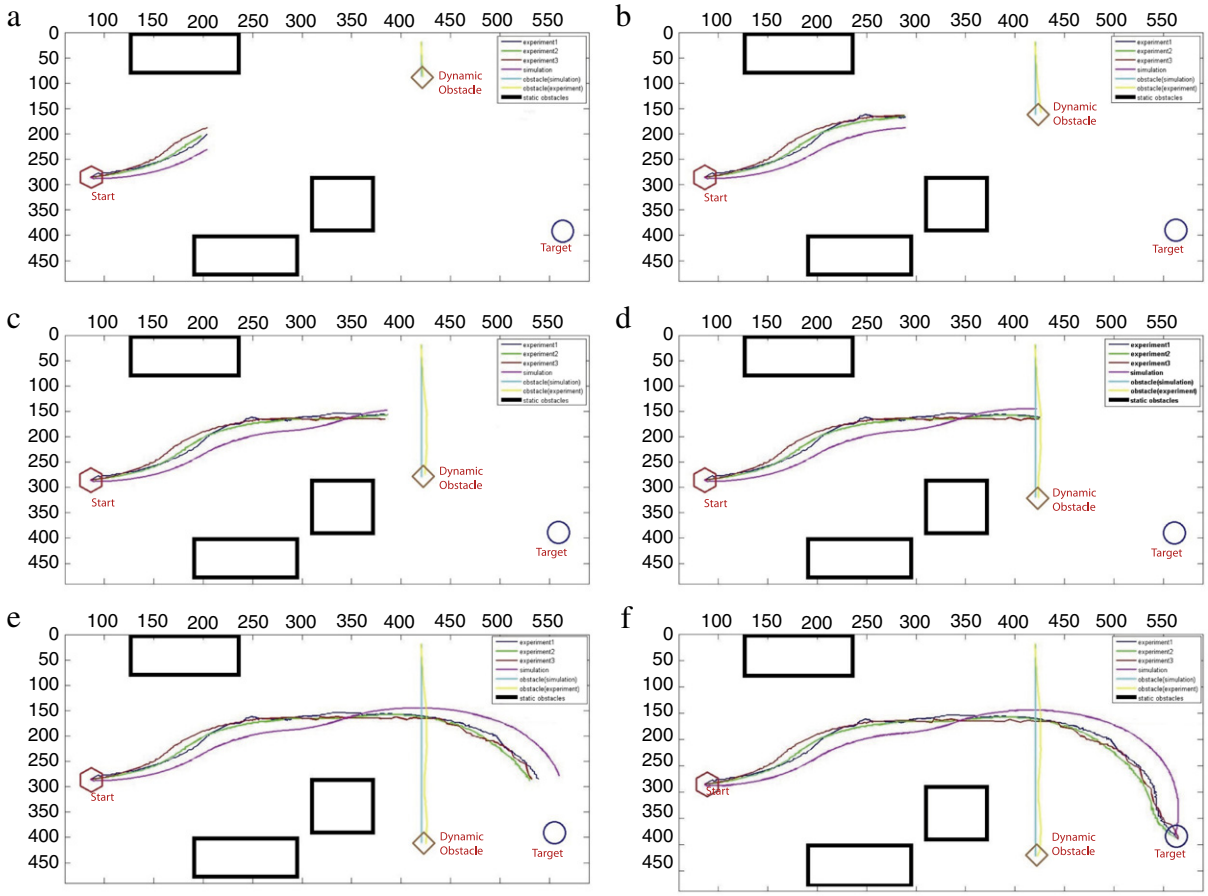


Fig. 21. Experimental evaluation of GAPCNN.

in a sinusoidal manner. Robot (pursuer) moves towards right as long as the target is moving right as shown in Fig. 17(a)–(c) and drastically readjusts its direction towards left as target start moving in that direction (Fig. 17(d)). Fig. 17(e) shows the pursuer following the target until an intercept is achieved.

8.2. Experimental results

In order to validate the results of the path planner, experiments were conducted using a P3AT mobile robot. An overhanging camera (Microsoft Kinect) was used as an image acquisition source. Using the procedure specified in Section 7, robot localization and navigation was made possible. To verify the effectiveness of the proposed modifications, a large number of experiments were conducted both in static as well as dynamic environments under identical conditions, a small subset of the results is presented in Table 3 and Figs. 18–20. In case of static as well as dynamic environments, the planner re-plans automatically at a rate of 5 Hz. This is to compensate for the inherent uncertainties in the robot motion in addition to the re-planning needed to ensure dynamic obstacle avoidance. Each time re-planning is carried out, state of the robot is updated and vision system acts as a feedback sensor.

8.2.1. Static environment

In case of the static experiment we used three static obstacles. The path followed by the robot in the three experiments conducted under identical conditions is shown in Fig. 19 while the path planned by GAPCNN at different instants while approaching the target location are shown in Fig. 18. A quick comparison of the paths taken by the robot during the experiments shows that its behavior is very close to the simulated behavior under identical conditions. The slight deviation from the simulation is due the stochastic nature of real robot dynamics.

8.2.2. Dynamic environment

The real effectiveness of the planner is marked by its performance in the dynamic environment. In this experiment three static obstacles and one dynamic obstacle, which intercepted the robot path, was used. In Fig. 20 "R" represents the robot, "D" is the dynamic robot while target is denoted by "T". The initial path planned by the GAPCNN planner is shown in Fig. 20(a). Until the time instant at which the robot is at the location shown in Fig. 21(a), the path planned by the planner is not intercepted by the dynamic obstacle (Fig. 20(b)). In the next time step (Fig. 21(b)) the planner shifts its optimal path to the top of the dynamic obstacle since the

Table 3
Path planning experiments.

Exp. No.	Complexity	Robot speed (m/s)	Obstacle speed (m/s)	Robot turnrate (degrees)
1	Three static obstacles	0.2	–	15
2	Three static and one dynamic obstacle	0.3	0.1	25

previously generated path becomes blocked by the dynamic obstacle (Fig. 20(c)). However when robot reaches the location as given in Fig. 21(c), the dynamic obstacles moves away, therefore the planned path shift downwards as shown in Fig. 20(d). In Fig. 21(d) and (e), the paths planned by the planner are largely straight to the goal but the robot takes a curvilinear trajectory because of its dynamics. The complete trajectory taken by the robot to its destination is shown in Fig. 21(f).

The repeatability in the paths taken by the robot in these experiments validates our approach. Lack of smoothness in the experimental curves as compared to that of simulation curve is largely because of uncertainty involved in the robot location estimation by the vision system. Also the deviation of the experimental curves from that of the simulation is that of the computational and communication overheads.

9. Conclusion

A novel approach based on a modified PCNN is presented for real time path planning and obstacle avoidance of mobile robots. Guided and adaptive behavior of the wave has resulted in significant improvement in path query time. Time efficiency of the algorithm proves that it can be used for path planning of static as well as dynamic environments. Simulation and experimental results have shown that the GAPCNN has a significant time improvement from that of the MPCNN while retaining all of its capabilities. It is important to note that the parameters used in the GAPCNN model are independent of the configuration space and a property of a neuron. These parameters shall work irrespective of the environment in which path is being planned.

Research is being carried out to extend the algorithm to include environments with uncertain robot, target and barrier locations. In addition to this dynamics constraints shall be introduced to the detected paths, thereby facilitating the motion of mobile machine.

Acknowledgment

The authors would like to thank Robotics and Artificial Intelligence department, NUST for their support in providing us with the facilities for hardware implementation. In particular the authors would like to mention the help, guidance and support extended by Mr. Muhammad Ali Ramay and Dr. Yasar Ayaz.

References

[1] J.A. Janet, R.C. Luo, M.G. Kay, The essential visibility graph: an approach to global motion planning for autonomous mobile robots, in: Proceedings, 1995 IEEE International Conference on Robotics and Automation, vol. 2, 1995, pp. 1958–1963.
[2] Han-Pang Huang, Shu-Yun Chung, Dynamic visibility graph for path planning, in: Proceedings, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004, vol. 3, 2004, pp. 2813–2818.
[3] F. Lingelbach, Path planning for mobile manipulation using probabilistic cell decomposition, in: Proceedings, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004, vol. 3, 2004, pp. 2807–2812.
[4] O. Takahashi, R.J. Schilling, Motion planning in a plane using generalized Voronoi diagrams, IEEE Trans. Robot. Automat. 5 (2) (1989) 143–150.
[5] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Proceedings, 1985 IEEE International Conference on Robotics and Automation, vol. 2, 1985, pp. 500–505.
[6] S.S. Ge, Y.J. Cui, Dynamic motion planning for mobile robots using potential field method, Auton. Robots 13 (2002) 207–222. http://dx.doi.org/10.1023/A:1020564024509.
[7] J.R. Andrews, Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator, Massachusetts Institute of Technology, Department of Mechanical Engineering, 1983.
[8] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Automat. 12 (4) (1996) 566–580.
[9] Yuandong Yang, Oliver Brock, Elastic roadmapsmotion generation for autonomous mobile manipulation, Auton. Robots 28 (2010) 113–130. http://dx.doi.org/10.1007/s10514-009-9151-x.
[10] M. Kalisiak, M. van de Panne, Rrt-blossom: Rrt with a local flood-fill behavior. in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006, ICRA 2006, pp. 1237–1242.
[11] J.J. Kuffner Jr., S.M. LaValle, Rrt-connect: an efficient approach to single-query path planning, in: Proceedings, IEEE International Conference on Robotics and Automation, ICRA '00, vol. 2, 2000, pp. 995–1001.
[12] S. Karaman, E. Frazzoli, Optimal kinodynamic motion planning using incremental sampling-based methods. in: 2010 49th IEEE Conference on Decision and Control (CDC), 2010, pp. 7681–7687.
[13] Sertac Karaman, Emilio Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894.
[14] Junfeng Yao, Chao Lin, Xiaobiao Xie, A.J. Wang, Chih-Cheng Hung, Path planning for virtual human motion using improved A* star algorithm. in: 2010 Seventh International Conference on Information Technology: New Generations (ITNG), 2010, pp. 1154–1158.
[15] Yanrong Hu, S.X. Yang, A knowledge based genetic algorithm for path planning of a mobile robot, in: Proceedings, 2004 IEEE International Conference on Robotics and Automation, ICRA '04, vol. 5, 2004, pp. 4350–4355.
[16] Soh Chin Yun, V. Ganapathy, Lim Ooi Chong, Improved genetic algorithms based optimum path planning for mobile robot. in: 2010 11th International Conference on Control Automation Robotics Vision, ICARCV, 2010, pp. 1565–1570.
[17] Xiaoyu Yang, M. Moallem, R.V. Patel, A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation, IEEE Trans. Syst. Man Cybern. B 35 (6) (2005) 1214–1224.
[18] P.I. Ro, B.R. Lee, Neural-fuzzy hybrid system for mobile robot path-planning in a partially known environment, in: Proceedings of the American Control Conference, vol. 1, 1995, pp. 673–677.
[19] R. Glasius, A. Komoda, S. Gielen, Population coding in a neural net for trajectory formation, Netw., Comput. Neural Syst. 5 (1994) 549–563.
[20] H.J. Caulfield, J.M. Kinser, Finding the shortest path in the shortest time using PCNN's, IEEE Trans. Neural Netw. 10 (3) (1999) 604–606.
[21] Hong Qu, S.X. Yang, A.R. Willms, Zhang Yi, Real-time robot path planning based on a modified pulse-coupled neural network model, IEEE Trans. Neural Netw. 20 (11) (2009) 1724–1739.
[22] L. Li, H. Ogmen, Visual guided motor control: adaptive sensorimotor mapping with on-line visual-error correction, in: World Congress Neural Network, 1993, 1994, pp. 127–134.
[23] P. Gaudiano, F. Muniz, E. Zalama, J. Lopez-Coronado, Neural Controller for a Mobile Robot in a Nonstationary Environment, Technical Report, Boston University, 1995.
[24] T. Fujii, Y. Arai, H. Asama, I. Endo, Multilayered reinforcement learning for complicated collision avoidance problems, in: Proceedings, 1998 IEEE International Conference on Robotics and Automation, vol. 3, 1998, pp. 2186–2191.
[25] I. Gavrilut, A. Gacsadi, C. Grava, V. Tiponut, Vision based algorithm for path planning of a mobile robot by using cellular neural networks, in: Robotics, 2006 IEEE International Conference on Automation, Quality and Testing, vol. 2, 2006, pp. 306–311.
[26] Fan Jian, Fei MinRui, Ma ShiWei, Rl-art2 neural network based mobile robot path planning, in: Sixth International Conference on Intelligent Systems Design and Applications, ISDA '06, vol. 2, 2006, pp. 581–585.
[27] R. Eckhorn, H.J. Reitboeck, M. Arndt, P. Dicke, Feature linking via synchronization among distributed assemblies: simulations of results from cat visual cortex, Neural Comput. 2 (3) (1990) 293–307.
[28] John L. Johnson, Dieter Ritter, Observation of periodic waves in a pulse-coupled neural network, Opt. Lett. 18 (15) (1993) 1253–1255.
[29] J. L. Johnson, Waves in pulse coupled neural networks, in: World Congress Neural Network, 1993, pp. 299–302.
[30] John L. Johnson, Pulse-coupled neural nets: translation, rotation, scale, distortion, and intensity signal invariance for images, Appl. Opt. 33 (26) (1994) 6239–6253.
[31] T. Lindblad, J.M. Kinser, Image processing using pulse-coupled neural networks, in: Perspectives in Neural Computing, Springer, 2005.
[32] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[33] Subhrajit Bhattacharya, Yagsbpl: A template-based c++ library for large-scale graph search and planning, 2013. Available at http://subhrajit.net/index.php?WPage=yagsbpl.

[34] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of robot motion: theory, algorithms, and implementations, in: Intelligent Robotics and Autonomous Agents, MIT Press, 2005.
[35] Eduard Oks, Micha Sharir, Minkowski sums of monotone and general simple polygons, Discrete Comput. Geom. 35 (2006) 223–240. http://dx.doi.org/10.1007/s00454-005-1206-y.
[36] Brian P. Gerkey, Richard T. Vaughan, Andrew Howard, The player/stage project: Tools for multi-robot and distributed sensor systems. in: Proceedings of the 11th International Conference on Advanced Robotics, 2003, pp. 317–323.

Usman Ahmed Syed received his B.E. in Mechatronics Engineering from National University of Sciences and Technology (NUST), Pakistan in 2011. Currently, he is a research associate in Probabilistic Robotics and Intelligent Systems in Motion (PRISM) Lab at Mechatronics Engineering department, NUST. His research interest includes computer vision, machine learning, motion planning, multi-agent systems and probabilistic domains for decision making under uncertainty.

Faraz Kunwar received his B.Sc. in Mechanical Engineering from College of E & ME, National University of Science and Technology (NUST Pakistan) and M.A.Sc./Ph.D. degrees from the University of Toronto (Canada) in 2005 and 2008, respectively. He currently directs the Probabilistic Robotics and Intelligent Systems in Motion (PRISM) Lab in the Department of Mechatronics in College of E & ME, National University of Science and Technology (NUST Pakistan). His research interests include guidance theory, robot navigation in realistic terrains and mobile robotics. He is the recipient of the Teachers and Researchers Overseas Scholarship Award from the Ministry of Science and Technology Pakistan.

Mazhar Iqbal is a faculty member at the Department of Basic Sciences and Humanities, College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan. He earned his Ph.D. in applied mathematics from Bahauddin Zakariya University Multan Pakistan in 2008. His research interests include Numerical Analysis, and computational methods.