International Journal of Robotics and Automation (IJRA)
Vol. 4, No. 1, March 2015, pp. 31~40
ISSN: 2089-4856

A PSO-Optimized Reciprocal Velocity Obstacles Algorithm for Navigation of Multiple Mobile Robots

Ziyad T. Allawi*, Turki Y. Abdalla**
*College of Education for Humanitarian Studies, University of Baghdad, Iraq
**Department of Computer Engineering, College of Engineering, University of Basrah, Iraq

Article Info

Article history:

Received May 31, 2014
Revised Sep 21, 2014
Accepted Oct 15, 2014

Keyword:

Multiple Mobile Robots
Navigation
Particle Swarm Optimization
Reciprocal Velocity Obstacles

ABSTRACT

In this paper, a new optimization method for the Reciprocal Velocity Obstacles (RVO) is proposed. It uses the well-known Particle Swarm Optimization (PSO) for navigation control of multiple mobile robots with kinematic constraints. The RVO is used for collision avoidance between the robots, while PSO is used to choose the best path for the robot maneuver to avoid colliding with other robots and to get to its goal faster. This method was applied on 24 mobile robots facing each other. Simulation results have shown that this method outperforms the ordinary RVO when the path is heuristically chosen.

Corresponding Author:

Ziyad T. Allawi,
College of Education for Humanitarian Studies
University of Baghdad, Iraq
Email: ziyad.allawi@gmail.com

1. INTRODUCTION

The robotic technologies have been widely employed in many applications. Nowadays, robot systems have been applied in factory automation, entertainment, space, etc... Recently, more and more researchers takes interest in the robot which can help people in our daily life, such as service robot, office robot, security robot, home robot, and so on.

Other than control and coordination of multiple mobile robots, one of the central problems in this area is motion planning among multiple moving mobile robots. In this paper, we address the problem of real-time navigation for multi-robot motion planning in dynamic environments. Each robot navigates independently without explicit communication with the other mobile robots. Therefore, we can formulate the basic problem as navigating a single agent to its goal location without colliding with the obstacles and the other mobile robots in the environment [1].

The problem of computing a collision-free path for a robot moving among dynamic obstacles is not only of interest to robotics but also has been widely studied for crowd simulation in computer graphics, virtual environments, video gaming, traffic engineering and architecture design, where each agent can be considered as a virtual human, a moving car, or an individual pedestrian [1]. It is important in many robotics applications, including automated transportation systems, automated factories, and applications involving robot human interactions, such as robotic wheelchairs.

The problem of collision avoidance between the robots is harder than the moving obstacles because the robots are not simply moving without considering their environment; they are also intelligent decision-making entities that try to avoid collisions as well. Simply considering them as moving obstacles may lead to *oscillations* if the other entity considers all other robots as moving obstacles as well. Therefore, the reactive nature of the other entities must be specifically taken into account in order to guarantee that collisions are

avoided [14]. An alternative to complete planning is to plan for the robot as it acts, taking new sensor inputs as they arrive and planning locally. Some of the prominent work in this area is based on the Velocity Obstacles (VO) [1].

This work introduces an optimized navigation method that combines the *reciprocal velocity obstacles* (RVO) [1] collision avoidance navigation method with the *particle swarm optimization* algorithm (PSO) [2]. RVO constructs the velocity obstacle regions for a given robot induced by other robots and chooses feasible velocity and orientation intervals considering the kinematic constraints of the robot. PSO inspects these intervals and chooses the best velocity and orientation that ensures a collision-free path for the robot which makes it closer to the desired goal. This method is simulated on 24 mobile robots facing each other; each robot tries to reach a goal that is behind its opposite partner. This method is compared with the same RVO method with arbitrary chosen configurations and the results were obtained for different simulation parameters.

The paper is structured as follows: Section 2 presents the related works of the navigation of multiple mobile robots; Section 3 illustrates the principles of the RVO and for the PSO. Section 4 shows the design procedure of the method with the simulation results; and finally, Section 5 holds the conclusion of the overall work.

2. RELATED WORKS

Numerous motion planning algorithms have been developed for mobile robots in static environments. In [3], the notion of a car-like robot was formalized, and the fact that a path for a holonomic robot lying fully in open regions of the configuration space can always be transformed into a feasible path for a nonholonomic robot was proven. Laumond *et al.* [3] also provided an algorithm to generate a feasible path for a nonholonomic robot from a path found for a holonomic robot. Approaches applicable to mobile robots have been developed for complete trajectory planning among moving obstacles, such as [4], [5].

Several variations of the velocity obstacle formulation have been proposed for multi-robot systems, generally by attempting to incorporate the reactive behavior of the other entities in the environment. Variations such as RVO [1], [21], recursive probabilistic velocity obstacles [22], [23], and common velocity obstacles [24] use various means to handle reciprocity, but each have their own shortcomings. Specifically, the approach of [23] may fail to converge, while other concepts [1], [24] are limited to dealing with only two robots.

Other work has focused mainly on follow-the-leader behavior when navigating robots in real-world settings [25], [26]. Also, there is a large body of work on centrally coordinating the motions of multiple robots [27]. However, there is little work on navigation of multiple independent robots to arbitrary goals in real world settings while taking into account the reactive behavior of other robots.

In the field of mobile robotics, many papers discussed the integration of PSO for control of mobile robot navigation. For instance, Vahdat *et al.* [6] proposed two recent sample-based evolutionary algorithms for global localization of a mobile robot. The first is the DE algorithm and the second is the PSO algorithm. They used PSO to adjust the location and velocity of the robot's pose. Juang and Chang [7] proposed an evolutionary-group-based particle-swarm-optimization (EGPSO) algorithm for fuzzy controller design. The objective of EGPSO was to improve fuzzy-control accuracy and design efficiency. EGPSO-designed fuzzy controller was applied to mobile-robot navigation in unknown environments. Kwok *et al.* [8] proposed a generic control structure based on the leader-follower strategy and virtual robot tracking framework to parameterize the formation configuration for cooperatively deploying the mobile robots into desired patterns. The PSO algorithm was used for its computationally-efficient capability of handling multi-objective criteria. Tang and Eberhard [9] presented an algorithm called augmented Lagrangian particle swarm optimization with velocity limits (VL-ALPSO). It used a PSO based algorithm to optimize the motion planning for swarm mobile robots. The algorithm combined the method of augmented Lagrangian multipliers and strategies of velocity limits and virtual detectors so as to ensure enforcement of constraints, obstacle avoidance and mutual collision avoidance. Yang and Li [10] proposed a new algorithm based on improved PSO of cubic splines for multiple mobile robot path planning. The center path was described by string of cubic splines, thus the path planning was equivalent to parameter optimization of particular cubic splines. Improved PSO was introduced to get the optimal path for the mobile robots. Mohamed et al. [11] describes a mathematical model for developing a scheme for an autonomous low cost mobile robot system using visual simultaneous localization and mapping and particle swarm intelligent path planner. The results indicated that this system could provide a solution for the problem of indoor mobile robot navigation.

3. RESEARCH METHOD

3.1. Reciprocal Velocity Obstacles

It is a navigation method used for robot motion planning in dynamic environments. It consists of selecting avoidance maneuvers to avoid static and moving obstacles in the velocity space, based on the current positions and velocities of the robot and obstacles. It is a first-order method since it does not integrate velocities to yield positions as functions of time [12].

The avoidance maneuvers are generated by selecting robot velocities outside of the VO, which represent the set of robot velocities that would result in a collision with a given obstacle that moves at a given velocity at some future time. To ensure that the avoidance maneuver is dynamically feasible, the set of avoidance velocities are intersected with the set of admissible velocities, defined by the robot's kinematics constraints.

Fiorini and Shiller [12] were the first who introduced this approach to be used in the path planning and navigation of mobile robots in dynamic environments which comprise the presence of static and moving obstacles (or other moving robots). They applied the approach upon the intelligent vehicles negotiating highway traffic.

The key features of the VO as mentioned in [12] were:

1. VO provides a simple geometric representation of potential avoidance maneuvers;
2. Any numbers of moving obstacles can be avoided by considering of the union of their VO's;
3. It unifies the avoidance of moving as well as stationary obstacles; and
4. It allows for the simple consideration of robot dynamics.

Since the VO had been introduced, some developments were carried out upon the original algorithm. Jur Berg contributed most of these developments. Reciprocal VO was introduced by Berg *et al.* [1], generalized VO was proposed by Wilkie *et al.* [13] and hybrid reciprocal VO was proposed by Snape *et al.* [14 and 15]. All these approaches were applied on the navigation of multiple mobile robots and it was mainly used for inter-robot collision avoidance. For instance, a hybrid reciprocal VO for collision-free and oscillation-free navigation of multiple mobile robots was presented in [15]. Each robot senses its surroundings and acts independently without central coordination or communication with other robots. The approach used both the current position and the velocity of other robots to compute their future trajectories in order to avoid collisions. The approach was reciprocal and avoids oscillations by explicitly taking into account that the other robots sense their surroundings as well and change their trajectories accordingly.

The method presented in this paper simultaneously determines actions for many robots that each may have different objectives. The actions are computed for each robot independently without communication among the robots. The method uses PSO algorithm to obtain the optimum collision-free velocity which guarantees the collision-free motion for each robot in the environment.

The fundamental configuration of the original VO is shown in the figure below [13]:

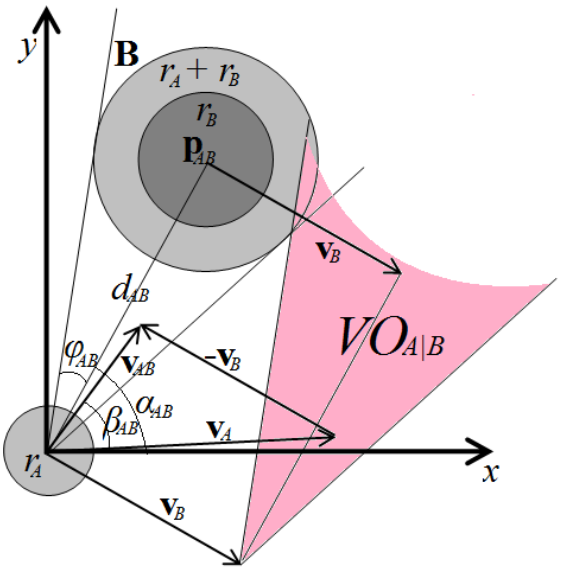

Figure 1. The Velocity Obstacle of Robot A induced by Robot B.

In the Figure 1, for a disc-shaped robot A (light gray) of radius r_A, located at $\mathbf{p}_A$, has a velocity of $\mathbf{v}_A$ and a disc-shaped robot B (dark gray) of radius r_B, located at $\mathbf{p}_B$, has a velocity of $\mathbf{v}_B$, the velocity obstacle for A induced by B, denoted $VO_{A|B}$ (the pink-colored cone), is the set of all velocities for A that would, at some point in the future, result a collision with B. This set is defined geometrically. Let the robot A be a single point in the origin and $\mathbf{B}$ be a disc (light gray ring encircling dark gray disc) centered at $\mathbf{p}_{AB}$ with a radius equal to the sum of A's and B's ($r_A + r_B$). If B is static (i.e. not moving), we could define a cone of velocities for A that would lead to a collision with B as the set of rays shot from the origin that intersect the boundary of $\mathbf{B}$. To derive a velocity obstacle from this, we simply translate the cone by the velocity $\mathbf{v}_B$ of B, as shown in figure. Briefly, the general representation of the Standard Velocity Obstacles appears in the Eq. 1 [13]:

$$VO_{A|B} = \{\mathbf{v} \mid \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - \mathbf{v}_B) \in \mathbf{B}\} \tag{1}$$

The Reciprocal Velocity Obstacles (RVO) originates from the mutual behavior of the two robots to avoid oscillation which may happen due to the change of direction. In this method, instead of choosing a new velocity for each robot that is outside the other robot's VO, we choose a new velocity that is the average of its current velocity and a velocity that lies outside the other robot's VO.

The general representation of the Reciprocal Velocity Obstacles (RVO) is shown below [1]:

$$RVO_{A|B} = \{\mathbf{v} \mid \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - (1-\alpha) * \mathbf{v}_A - \alpha * \mathbf{v}_B) \in \mathbf{B}\} \tag{2}$$

where α is a constant which refer to the effort share between the robots. In the standard RVO, α=0.5.

This method performs simpler calculations than the original RVO in [1] using the shape of RVO cone (base angle and axis) to check if $\mathbf{v}_A$ is inside the cone or outside it; and permits the use of optimization algorithms.

The algorithm begins by calculating the distance between the two robots d_{AB} and its argument α_{AB} with respect to the Cartesian coordinates as in Eq. 3:

$$\begin{aligned} \mathbf{p}_{AB} &= \mathbf{p}_B - \mathbf{p}_A = [x, y] \\ d_{AB} &= \|\mathbf{p}_{AB}\| = \sqrt{x^2 + y^2} \\ \alpha_{AB} &= \text{atan2}(y, x) \end{aligned} \tag{3}$$

The line d_{AB} and its argument α_{AB} represent the cone axis of symmetry. Then, the cone base semi-angle φ_{AB} can be calculated as shown in the Eq. 4:

$$\varphi_{AB} = \sin^{-1}\left(\frac{r_A + r_B}{d_{AB}}\right) \tag{4}$$

The parameters $\mathbf{v}_B$, d_{AB}, α_{AB}, and φ_{AB} can be used to check the robot A's possibility of collision with robot B as follows. We assume $\mathbf{v}'_A$ to be a heuristically chosen velocity of the robot A, subjected to the nonholonomic constraints of that robot. First, the magnitude of velocity difference between the two robots and its argument with respect to the Cartesian coordinates, v_{AB} and β_{AB}, respectively are calculated as shown in Eq. 5:

$$\begin{aligned} \mathbf{v}_{AB} &= (\mathbf{v}'_A - \tfrac{\mathbf{v}_A + \mathbf{v}_B}{2}) = [x, y] \\ v_{AB} &= \|\mathbf{v}_{AB}\| = \sqrt{x^2 + y^2} \\ \beta_{AB} &= \text{atan2}(y, x) \end{aligned} \tag{5}$$

The sufficient condition of being $\mathbf{v}'_A$ inside the $RVO_{A|B}$ is when ψ_{AB}, the absolute difference between α_{AB} and β_{AB}, is smaller than φ_{AB} as in Eq. 6:

$$\begin{aligned} &\psi_{AB} = |\beta_{AB} - \alpha_{AB}| \\ &\begin{cases} \mathbf{v}'_A \in RVO_{A|B} & \text{if } \psi_{AB} \le \varphi_{AB} \\ \mathbf{v}'_A \notin RVO_{A|B} & \text{if } \psi_{AB} > \varphi_{AB} \end{cases} \end{aligned} \tag{6}$$

If $\mathbf{v'}_A$ lies in the $RVO_{A|B}$, then changing the magnitude and/or the direction of $\mathbf{v'}_A$ is mandatory for escaping collision. Choosing $\mathbf{v}_A^*$ (optimum velocity) for escaping from all the robots in the environment may be carried out by using heuristic methods or optimization techniques considering subjection to the nonholonomic constraints of the robot A in linear and angular velocities.

One can calculate the minimum possible time of collision (if there is one). This time is calculated only if $\mathbf{v'}_A$ lies in the $RVO_{A|B}$, because otherwise the time will be infinite. Eq. 7 explains:

$$t_c = \frac{d_{AB}\cos\psi_{AB} - \sqrt{(r_A + r_B)^2 - d_{AB}^2 \sin^2\psi_{AB}}}{v_{AB}} \tag{7}$$

The term under the square root will be positive only if $\psi_{AB} \leq \varphi_{AB}$.

The time of collision t_c should be compared with the sampling time of the simulation τ. Although $\mathbf{v}_A$ lies in the $VO_{A|B}$, but the collision will be certain only when τ is greater than t_c. If the opposite happen (i.e. τ is smaller than t_c), the collision will not happen in the next time sample although the collision is still possible, then a penalty formula can be used to choose the best velocity among a set of candidate velocities. The penalty formula used in this work is shown in Eq. 8 [1]:

$$p(\mathbf{v'}_A) = \frac{k}{t_c} + \left\| \mathbf{v}_A^G - \mathbf{v'}_A \right\| \tag{8}$$

where $\mathbf{v}^G{}_A$ is the goal-directed velocity of the robot A and k is a constant. The penalty is increased when the robot velocity diverges from the goal velocity and the collision time is short. The optimum velocity $\mathbf{v}_A^*$ will be the velocity which has the least penalty (i.e. close to the goal velocity and out from the RVO).

This algorithm may be used for optimization concerns where the optimization algorithm should use these previous assumptions to select another velocity heuristically until an optimum velocity $\mathbf{v}_A^*$ is found which guarantees a collision-free path for the robot A in the overall environment.

The Reciprocal Velocity Obstacles pseudo algorithm is shown below:

Function V=*VelocityObstacles*(*n*)
// *n*=number of robots in the environment
for *i*=1 **to** *n* **do**
$\mathbf{p}_i$=position of robot *i* [x_i, y_i, θ_i]; // θ_i may be directed toward a target
$\mathbf{v}_i = v_{i\max}$*[cosθ_i, sinθ_i];
r_i=radius of the robot;
end for
for *i*=1 **to** *n* **do**
for *j*=1 **to** *n*, $j \neq i$ **do**
d_{ij} & α_{ij}=the RVO cone axis and its argument as in Eq. 3;
φ_{ij}=the RVO cone semi-angle as in Eq. 4;
end for
loop
f=0;
for *j*=1 **to** *n*, $j \neq i$ **do**
v_{ij} & β_{ij}=the magnitude and argument of velocity difference as in Eq. 5;
Use Eq. 6 to check if $\mathbf{v}_i$ lies in the $RVO_{A|B}$; if so, *flag*=1; else *flag*=0;
t_c=collision time (if $\mathbf{v}_i$ lies in the $RVO_{A|B}$) as in Eq. 7;
Check if $t_c > \tau$; if so, *flag*=0;
f=*f* + *flag*;
end for
Use heuristics or an optimization method to find $\mathbf{v}_i^*$ that minimizes *f* considering the nonholonomic constraints of the robot *i* or by using Eq. 8;
until $\mathbf{v}_i^*$ is found;
$\mathbf{V}_i = \mathbf{v}_i^*$;
end for
return V;
end.

3.2. Particle Swarm Optimization

Particle Swarm Optimization Algorithm is one of the widely used bio-inspired optimization algorithms which include Ant Colony Optimization (ACO), Differential Evolution (DE) and Artificial Bee Colony (ABC). PSO uses the idea of the social behavior for movement of flock of birds or school of fish.

PSO is a meta-heuristic algorithm, i.e. it makes few or no assumptions about the problem being optimized and does not guarantee an optimal solution is ever found.

PSO was originally introduced by Kennedy and Eberhart [16] in 1995. PSO was primarily developed for optimizing continuous nonlinear functions and systems. Some paradigms that implement the concept of particle swarm were demonstrated, and then one of these paradigms was implemented in details, then the new algorithm was tested for neural network weight training. In the very same year, Kennedy and Eberhart [17] developed another two paradigms for PSO and integrated them in the same algorithm to make it more powerful. They used local and global oriented search method for finding the optimum value for a single particle and for the particles as a whole. These two paradigms improved the algorithm greater than the first proposed one.

Another touch on the original PSO was from Shi and Eberhart [2] when they proposed a modified optimizer for the algorithm. A new parameter called (Inertia Weight) was introduced in the original algorithm to make balance between global and local search. This parameter could be positive constant or positive linear or nonlinear function of time.

PSO has been used-since it was proposed – in a wide area of applications. Poli [18] listed 26 major applications which PSO was used within. Some of these applications were antennas, communications, control, fuzzy systems, graphics, neural networks, robotics, wireless sensor networks, signal processing etc…

The problems which face the PSO algorithm are limited to the slow convergence and trapping into local optimum values. Several modifications have been carried out upon PSO to overcome these problems. One of these modifications was proposed by Wang and Fan [19]. They proposed an improved PSO algorithm by nonlinearly decreasing the value of inertia weight. The simulation results revealed that the proposed PSO had the best convergence speed and reaching to optimum compared to the original algorithm.

As the most of the bio-inspired algorithms, PSO was originally used for global multi-dimensional optimization. It shares many similarities with evolutionary algorithms such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In the PSO algorithm, the birds in a flock are symbolically represented as “particles”. These particles can be considered as simple agents “flying” through a problem space. A particle’s location in the multi-dimensional problem space represents one solution for the problem. When a particle moves to a new location, a different problem solution is generated. This solution is evaluated by a fitness function that provides a quantitative value of the solution’s utility. Each individual or particle in a swarm behaves in a distributed way using its own intelligence and the collective or group intelligence of the swarm. As such, if one particle discovers a good path to food, the rest of the swarm will also be able to follow the good path instantly even if their location is far away in the swarm.

The advantage of using an optimization method such as PSO is that it does not rely explicitly on the gradient of the problem to be optimized, so the method can be readily employed for a host of optimization problems. This is especially useful when the gradient is too laborious or even impossible to derive.

In the context of multivariable optimization, the swarm is assumed to be of specified or fixed size with each particle located initially at random locations in the multidimensional design space. Each particle is assumed to have two characteristics: a position and a velocity. Each particle wanders around in the design space and remembers the best position (in terms of the food source or objective function value) it has discovered. The particles communicate information or good positions to each other and adjust their individual positions and velocities based on the information received on the good positions.

Although each bird has a limited intelligence by itself, it follows the following simple rules [20]:

1. It tries not to come too close to other birds.
2. It steers toward the average direction of other birds.
3. It tries to fit the “average position” between other birds with no wide gaps in the flock.

Thus the behavior of the flock or swarm is based on a combination of three simple factors:

1. Cohesion-stick together.
2. Separation-don't come too close.
3. Alignment-follow the general heading of the flock.

In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that

can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

PSO solves a problem as follows: first, the particles are initialized randomly. Then, it finds the best solution by iteration. Each particle flies through the solution space of problem and adjust its flying velocity to search for the global optimum according to its own and social historical experiences. Through the iteration, each particle updates itself by following the two best values, one of which is the best solution found by the particle itself, we call it personal best value, namely $\mathbf{P}_{best}$, and the other is the best solution found by the whole swarm, we call it global best value, namely $\mathbf{G}_{best}$. When the two best values are found, each particle updates its velocity and position according to the formula as follows [2, 20]:

$$\begin{aligned} \mathbf{v}_j(i+1) &= w \cdot \mathbf{v}_j(i) + c_1 \cdot r_1 \cdot (\mathbf{P}_{jbest}(i) - \mathbf{x}_j(i)) + c_2 \cdot r_2 \cdot (\mathbf{G}_{best}(i) - \mathbf{x}_j(i)) \\ \mathbf{x}_j(i+1) &= \mathbf{x}_j(i) + \mathbf{v}_j(i+1) \end{aligned} \tag{9}$$

where $\mathbf{v}_j(i)$ and $\mathbf{x}_j(i)$ are the n-dimensional velocity and position vectors of the j^{th} particle in the i^{th} iteration; r_1 and r_2 are two random functions in the range [0, 1], c_1 and c_2 refer to the two acceleration factors, usually belong to the interval [0, 4], and w is the inertial weight. For the standard PSO, the w is a constant. In the modified PSO by Shi and Eberhart [2] where they presented w, they proposed a linearly decreasing equation to calculate it, but they infer that it is possible to use any nonlinear decreasing equation to represent w. The inertial weight is defined as follows:

$$w(i) = w_{\min} + (w_{\max} - w_{\min}) \cdot \left(\frac{i_{\max} - i}{i_{\max}}\right)^2 \tag{10}$$

where $w_{\max}$ and $w_{\min}$ are the maximum and minimum values of the inertial weight, and $i_{\max}$ is the maximum possible number of iterations.

The constants c_1 and c_2 decide the relative influence of the social and cognition components. Typically these constants have the same value (c_1=c_2=2) to give equal weight to each social and cognition components. Also, the inertial weight w has two extremes, typically min. 0 and max. 1.

The PSO pseudo algorithm is shown below:

Function $\mathbf{G}_{best}$=*ParticleSwarm*(N, $i_{\max}$)
// N=number of particles, $i_{\max}$=maximum no. of iterations.
Initialize $\mathbf{x}_j(0)$, (j=1, 2, …, N) randomly, $\mathbf{x}_j \in \mathbb{R}^n$, $\mathbf{x}_{\min} \le \mathbf{x}_j \le \mathbf{x}_{\max}$;
i=0;
loop
Evaluate Fitness $f_i(i)$ for each particle $\mathbf{x}_j(i)$;
Determine $\mathbf{G}_{best}$, the particle which has the global best fitness;
If a stopping condition is satisfied, **break**;
Update the particles using Eq. 9;
i=i + 1;
until i=$i_{\max}$;
return $\mathbf{G}_{best}$;
end.

4. RESULTS AND DISCUSSIONS

The design procedure is done under MATLAB® environment by using 24 identical mobile robots. It will be assumed that a simplified robot model will be used, where each robot is assumed to have a simple shape (circle) moving in a two-dimensional workspace. Also, each robot has perfect sensing, and is able to infer the exact shape, position and velocity of other robots in the environment. The robots are positioned in a circle perimeter and facing its center as in Figure 2. The goals are located in the same place of their opponents. So, the robot will try to maneuver all other 23 robots to escape and reach its goal. For comparison, we used the RVO method with arbitrary chosen velocities.

In the simulation phase, the single particle is a 2-dimension vector holding the magnitude and direction of robot's velocity. The fitness equation used in simulation is Eq. 8, and the values of c_1 and c_2=2. We vary the penalty constant k and the population size N. The maximum number of iterations is fixed. Its values is $i_{\max}$=200. The minimum and maximum values of the inertia weight are 0 and 1, respectively.

The robot specifications are shown in Table 1.

Table 1. Specifications of the Mobile Robot

Specification	Value
Wheel Rotational Velocity	20 rad/s
Angular Steering Velocity	5 rad/s
Radius of the Robot	10 cm
Radius of the Wheel	5 cm
Maximum Heading Velocity	100 cm/s

Tables 2 and 3 illustrate the mean robot travelling distances (in cm) for the two methods respectively for various k and N (Direct Displacement=1000).

Table 2. Mean Distances for robots in RVO method

RVO	Distance
k=5	1133
k=10	1158
k=20	1207
k=50	1262

Table 3. Mean Distances for robots in PSO-RVO method

PSO-RVO	N=10	N=20	N=50	N=100
k=5	1140	1112	1103	1096
k=10	1155	1197	1180	1174
k=20	1200	1233	1221	1194
k=50	1300	1244	1266	1283

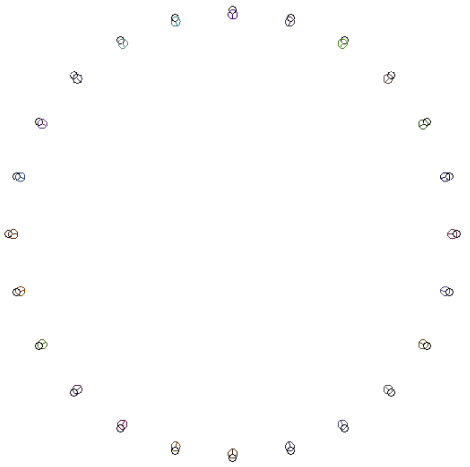

Figure 2. Robot Positions for PSO-RVO algorithm, N=100, k=5.

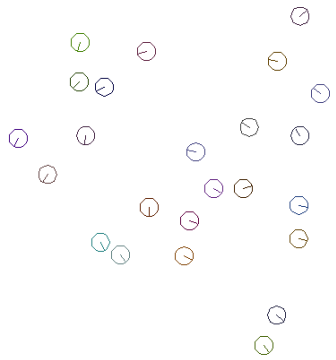

Figure 3. Robots are maneuvering each other

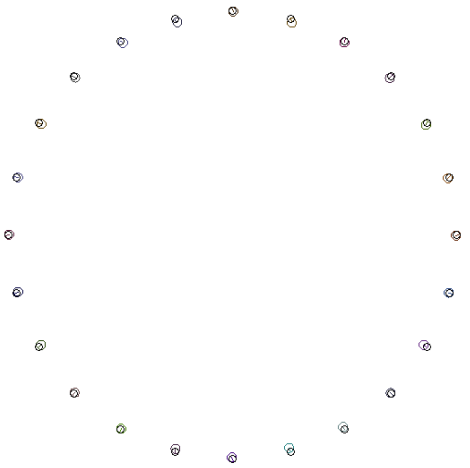

Figure 4. All robots are in the goals

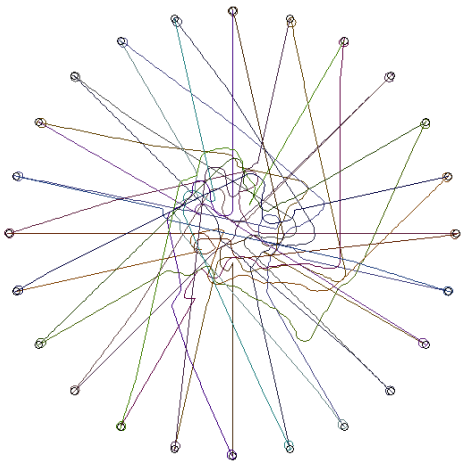

Figure 5. Robot Paths in the above scenario

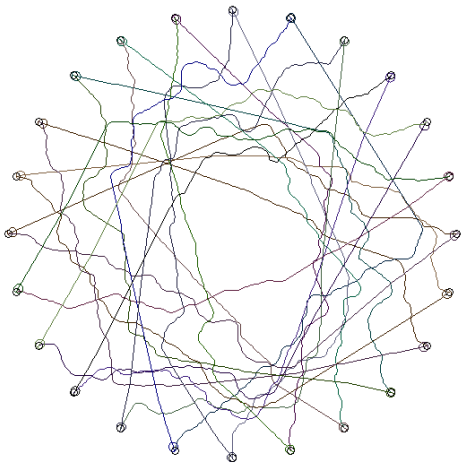

Figure 6. Robot Paths for RVO method, k=50

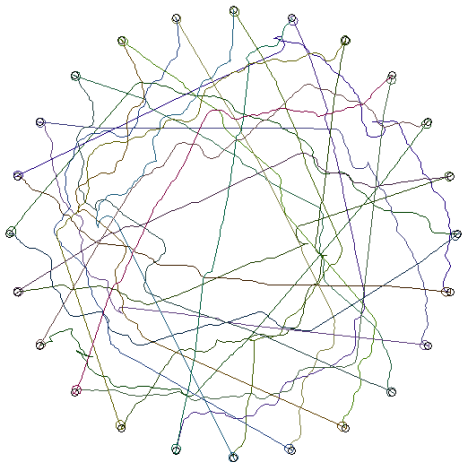

Figure 7. Robot Paths for PSO-RVO method for N=20, k=50

In the above Figures (2 to 5), we see the behavior of the robots in a scenario of facing each others with the parameters k=5 and N=100. In Figure 2, the robots are heading directly towards their goals despite of the possible collisions. The robots then begin to maneuver when the distances between them are being smaller and smaller. PSO tries to select the best next move for the robots depending on their current configuration and the other robots configuration. The maneuver takes place in a small area of the environment, that's because k is small. We see in Figure 3 (zoomed in) that some of the robots are close to each other but they will not collide because that PSO chooses the best collision free path for every one. The robots then succeeded to escape the maneuvering space and moved directly towards their destinations. In Figure 4, all the robots have arrived safely to their goals.

Figure 5 illustrates the paths which all robots take. We see that the maneuvering took place in a small area, and the robots moved in a pure line from their origins to the maneuvering space and from that space to the goals after escaping.

We see that the robot paths have reasonable oscillations in some regions of the environment, that's because k is small. Choosing k is very important in that situation; it should be small enough to cancel all oscillations that may happen in the course of maneuvering. But if one chooses k very small, the process may fail due to collisions before the maneuvering.

Figures 6 and 7 show the environment when choosing high k. It's clear that the robot paths are oscillatory from the beginning and do not be a direct path unless the robot escapes maneuvering. We see in the Figure 8 the oscillations are more than the oscillations in Figure 7, that's because of the arbitrary choice of velocities in ordinary RVO method.

If we look to the tables 2 & 3, we see that PSO-RVO outperforms itself and the ordinary RVO in the mean robot distances when k=5, and becomes more and more oscillatory with the increase in travelling distance when k is increasing; therefore, one should choose k wisely to ensure collision-free and oscillation-free navigation of multiple robots.

Increasing N aids in finding the optimal next move of the robot. We see in Table 3 that the minimum mean distance travelled by the robots occur when N=100, which is 1096 cm. As in all optimization algorithms, increasing agent size is important to find the optimal solution but it consumes time; therefore one should choose a moderate value for N if he wants to apply this algorithm in real-time environments.

5. CONCLUSION

It can be concluded from this work that using the optimization methods in the navigation of multiple mobile robots helped in someway to increase the efficiency of robot maneuvering. This is clear from the results shown in Tables 2 and 3. PSO-RVO method has succeeded in creating a collision-free, oscillation-free optimum path for all the robots provided that choosing the appropriate values of k and N.

REFERENCES

[1]. J. van den Berg, *et al.*, "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation", IEEE International Conference on Robotics and Automation, 2008, pp. 1928-1935.

[2]. Y. Shi and R. Eberhart, "A modified particle swarm optimizer", Proceedings of IEEE International Conference on Evolutionary Computation, 1998, pp. 69-73.

[3]. J.P. Laumond, *et al.*, "A motion planner for nonholonomic mobile robots", *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 5, 1994, pp. 577-593.

[4]. E. Frazzoli, *et al.* "Real-time motion planning for agile autonomous vehicles", *Journal of Guidance Control and Dynamics*, Vol. 25, No. 1, 2002, pp. 116-129.

[5]. D. Hsu, *et al.*, "Randomized kinodynamic motion planning with moving obstacles", *The International Journal of Robotics Research*, Vol. 21, No. 3, p. 233, 2002.

[6]. A. R. Vahdat, *et al.*, "Mobile Robot Global Localization using Differential Evolution and Particle Swarm Optimization", IEEE International Conference of Evolutionary Computation, 2007, pp. 1527-1534.

[7]. C. Juang, and Y. Chang, "Evolutionary-Group-Based Particle-Swarm-Optimized Fuzzy Controller with Application to Mobile-Robot Navigation in Unknown Environments", *IEEE Transactions on Fuzzy Systems*, Vol. 19, No. 2, 2011, pp. 379-392.

[8]. N. M. Kwok, *et al.*, "PSO-Based Cooperative Control of Multiple Mobile Robots in Parameter-Tuned Formations", Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering, 2007, pp. 332-337.

[9]. Q. Tang and P. Eberhard, "A PSO-based algorithm designed for a swarm of mobile robots", *Springer Journal of Structural and Multidisciplinary Optimization*, Vol. 44, No. 4, 2011, pp. 483-498.

[10]. M. Yang and C. Li, "Path Planning and Tracking for Multi-robot System Based on Improved PSO Algorithm", IEEE International Conference on Mechatronic Science, 2011, pp. 1667-1670.

[11]. A. Z. Mohamed, *et al.*, "A faster path planner using accelerated particle swarm optimization", *Springer International Journal of Artificial Life and Robotics*, Vol. 17, 2012, pp. 233-240.

[12]. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", *The International Journal of Robotics Research*, Vol. 17, No. 7, 1998, pp. 760-772.

[13]. D. Wilkie *et al.*, "Generalized Velocity Obstacles", The IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5573-5578.

[14]. J. Snape, *et al.*, "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5917-5922.

[15]. J. Snape, *et al.*, "The Hybrid Reciprocal Velocity Obstacle", *IEEE Transactions on Robotics*, Vol. 27, No. 4, 2011, pp. 696-706.

[16]. J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proceedings of IEEE 4th International Conference on Neural Networks, 1995, pp. 1942-1948.

[17]. R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory", IEEE 6th International Symposium on Micro Machine and Human Science, 1995, pp. 39-43.

[18]. R. Poli, "Analysis of the Publications on the Applications of Particle Swarm Optimization", Hindawi Journal of Artificial Evolution and Applications, 2008, pp. 1-10.

[19]. D. Wang and F. Fan, "Study on Modified PSO Algorithm", IEEE 6th International Conference on Natural Computation, 2010, pp. 640-645.

[20]. S. S. Rao, "Engineering Optimization", John Wiley & Sons Inc., Hoboken, New Jersey, USA, 2009.

[21]. S. J. Guy, *et al.*, "Clear Path: Highly parallel collision avoidance for multi-agent simulation," in Proceedings of the ACM SIGGRAPH Eurographics Symposium of Computer and Animation, 2009, pp. 177-187.

[22]. C. Fulgenzi, *et al.*, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid," in Proceedings of the IEEE International Conference of Robotics and Automation, 2007, pp. 1610-1616.

[23]. B. Kluge and E. Prassler, "Reflective navigation: Individual behaviors and group behaviors," in Proceedings of the IEEE International Conference of Robotics and Automation, 2004, pp. 4172–4177.

[24]. Y. Abe and M. Yoshiki, "Collision avoidance method for multiple autonomous mobile agents by implicit cooperation," in Proceedings of the IEEE RSJ International. Conference of Intelligent Robotic Systems, 2001, pp. 1207-1212.

[25]. K. C. Ng and M. M. Trivedi, "A neuro-fuzzy controller for mobile robot navigation and multirobot convoying," *IEEE Transactions of Systems, Man and Cybernetics B*, Vol. 28, No. 6, 1998, pp. 829-840.

[26]. S. Carpin and L. E. Parker, "Cooperative motion coordination amidst dynamic obstacles," in Proceedings of the International Symposium of Distributive Autonomous Robotic Systems, 2002, pp. 145-154.

[27]. S. M. LaValle, "Planning Algorithms", Cambridge, U.K. Cambridge University Press, 2006.