ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse32852019.pdf
https://doi.org/10.30534/ijatcse/2019/32852019

Circular Shape Formation using Self Adaptive Collective Motion of Swarm Robots

Ahmad al-Qerem
Zarqa University, Jordan, ahmad_qerm@zu.edu.jo

ABSTRACT

This paper present a method for self-organized shape formation swarm using collective motion algorithm and move them from source to destination along predefined path in a cohesive way. Collective motion is essential for swarm to keep connectivity between their members during the movement and in the proposed work it used to shape formation. Using the proposed self-adaptive collective motion swarm robots can organize their self in a circular shape and move from source to destination along predefined linear path while preserve their shape formation.

Key words: Collective motion, shape formation, Swarm intelligence, Topology Force, Guidance Force

1.INTRODUCTION

Swarm is a large set of homogenous agent interacting among themselves locally and with their environment to form a global emerge of collective behavior due to their perception in the neighborhood. Swarm-based algorithms have been introduced to provide low cost, fast, and robust solutions for especially real-world problems, by modeling the behaviors of numerous swarm of animals and insects such as ants, bees, birds, and fish [1][2]. Swarm Intelligence (SI) and bio-inspired computing based on different natural swarm systems and it was successfully applied in many real-life applications by simulating collective behavior of these natural swarms, and can be used in controlling robots and unmanned vehicles, predicting social behavior, enhancing the telecommunication and computer networks, in a variety fields in engineering, social, and sciences.

Swarm intelligence algorithms must be flexible to internal and external changes to be robust when some member fail, so the collective behavior is essential for swarm agents or robots for coordination of activities and travel directions [3], and this collective behavior can be divide to consensus (decision making), coordinate formation, pattern formation, collective motion, and synchrony [9], in this paper collective motion, and shape formation will be so these robots can travel from source to destination in cohesive way in predefined shape formation.

In artificial swarm intelligence global shape formation is one of the challenging problems. It is used for different purposes, such as save energy and path optimization. For instance, a flock of large bird fly forming a V shape to reduce the air resistance and fatigue for physically weak birds. Ants moving in a line to optimize the path to their food source by laying without any central control. For specific task-oriented performance in artificial intelligence systems such as exploring and mapping in space, building sensing grids, forming a barricade for protecting an area, Shape formation is required [4].

2.PREVIOUS WORK

Paper [5] (shaping formation) proposed an algorithm of rectilinear motion control and navigation of swarm of autonomous homogeneous mobile robots involved in the formation of a convex space surface, with a given packing density the convexity of the surface, as well as the absence of obstacles. The proposed algorithm consists of three main stages: (a) the analysis and clothing of the formed convex surface and calculation of the coordinates of target points and portal points; (b) the mapping between robots and target points; (c) calculation of the trajectories of robots taking into account possible collisions and the initial delay time of each robot, experimental robots number from 10 to10,000.

Paper [6] (shaping formation) discussed how a swarm of differential-drive robots can self-organize itself into multiple nested layers of a given circle, rectangular, triangle, cross shape. A key component of the proposed work is the reliance on inter-robot collisions to provide information on how the formation should grow. The number of robots in the swarm increases from tens to several hundred robots. The average quality of the formation is shown to be a linearly decreasing function of swarm size, and the time for a swarm to form a given shape does not grow quickly even as the number of robots in the swarm increases by a large amount. The proposed robot controller is composed of three subsystems. (a) The orbit and steering module which responsible for directing the robot towards an appropriate location on the perimeter of

the desired shape. (b) The collision avoidance subsystem slows the robot down when it detects a nearby obstacle, but does not steer away to avoid it. (c) The layer promotion subsystem that determines when a robot should "promote" itself to the next layer of the formation in order to avoid congestion and free up space for other robots. In this paper no robot motion is discussed just formation.

Paper [7] (collective motion) discussed collective dynamics of self-propelled mobile particles which able to probe and anticipate the orientation of their neighbors in three spatiotemporal patterns: homogeneous and coherent directed motion (flocking), synchronous circular motion (spinning) and compact group propagating in a coherent fashion (swarming).

3.PROBLEM STATEMENT

For many years ago swarm intelligence has been studied and deployed in many applications such as aircraft formation, theoretical research, and engineering application, due to its flexibility, scalability, and robustness. One of the most attractive and challenging topics in Swarm intelligence is formation control which has many application like aerospace, rescue missions, military affairs. Swarm control design is a complex field because of many issues that should be considered such as coordination between swarm robots, communication, and formation keeping [11].

There are three main methods to implement formation control (a) behavior based formation which utilized a set of predefined behavior parameter such as moving to target destination, obstacles avoidance, and formation keeping, its suitable to distributed strongly autonomous multi-robot system, but the stability of a desired formation is not guarantee when the environment is complex [12]. (b) Virtual structure in this method the robots is considered as some points in the system and defining the overall behavior formation structure, by tracking a given desired trajectory and knowing the desired behavior of the virtual structure, after that this behavior is converted to individual robots by imposing a certain formation shape, but because of its restricted requirements which needed to maintain the virtual structure formation makes this method application limited [13]. (c) And leader follower method by which a leader is chosen and each follower keep track with this leader and keep a certain distance and direction angle from him, but this method prone to robots collision deadlock [11].

Behavior based method is used in this paper by implementing collective motion algorithm so a swarm of robots are form a circular shape and moving in linear predefined path while the circular shape is preserved.

4.RELATED WORK

This paper is based on work and result concluded in paper [3] which discuss the collective motion of swarm robots and consider this type of motion as a fundamental operation of swarm to enable swarm move from source to destination in a cohesive way. Paper [3] presents Self adaptive collective motion algorithm for swarm robots in 3-d space, using one hop neighbor information and without centralization in control, self-adaptive this mean robots based on their environments can dynamically determine proper moving parameters, and enable robots to move from source to destination on predefined path considering the following requirements:

- Use information from only one neighbor hop.
- Maintain connectivity of the network topology to enable information exchange.
- Capability to bypass obstacles without cause swarm partitioning.
- Desired distance between neighboring is maintaining

This paper develop the first algorithm from paper [3] case one: "no obstacles or leader" to make swarm robots shape in circle and move in predefined path with shape keeping Guidance force $\overrightarrow{F_G}$ guide robots to their destination along the predefined path and ensure their continuous movement until reaching destination. Topology force $\overrightarrow{F_T}$ is used to maintain a good topology of the robot swarm by maintain the connectivity of the network topology and desired distance between neighbors. Both topology and guidance force are applied on each robot and make it move by the resultant force $\overrightarrow{F_{RS}}$.

The predefined path P is a sequence of coordinates P_i, where i = 1, 2, 3… k, and its known for each robot. A robot start from P_1 the source toward P_{i+1} and continue sequentially until it reach the destination P_k. When the robot move from its current location $c = \langle x_c, y_c \rangle$ to the next coordinate P_{i+1} the guidance force affecting the robot is defined by the following equation:

$$\overrightarrow{F_G} = \overrightarrow{\sqrt{(x_{i+1} - x_c)^2 + (y_{i+1} - y_c)^2}} \qquad (1)$$

The Euclidean distance between c and P_{i+1} is the magnitude of $\overrightarrow{F_G}$ and its direction is from c to P_{i+1}.

The topology force $\overrightarrow{F_T}$ between two neighbors is presented by two virtual forces, attractive and repulsive forces between these neighbors, and only one force acts at a given time. Let $D_{i,j}$ represent the distance between two neighbors i and j where $0 < D_{i,j} \leq R_c$, R_c is the communication range, and R_d is the desired distance that required between two neighbors .

Attractive force will take affects when $R_d < D_{i,j} \leq R_c$; otherwise, Repulsive force will take the affect ($D_{i,j} \leq R_d \leq R_c$) so if the distance between two neighbors is

greater than the desired distance then the two neighbors will be attractive to each other by the effect of the attraction force, in the other hand if the distance between two neighbors is smaller than the desired distance then the repulsive force will take the affect to avoid collision between neighbors.

$$\overrightarrow{f_A(\iota,J)} = \sqrt{(x_\iota - x_J)^2 + (y_\iota - y_J)^2} \quad (2)$$

$$\overrightarrow{f_R(\iota,J)} = \begin{cases} \overrightarrow{\frac{M}{D_{ij}^2} - \frac{M}{R_d^2}} & if\ 0 < D_{ij} < R_d \\ 0 & D_{ij} < R_d \end{cases} \quad (3)$$

Where $\overrightarrow{f_A(\iota,J)}$ and $\overrightarrow{f_R(\iota,J)}$ re attractive and repulsive force acting on robot i from its neighbor j, and M is a positive constantan and set to 10.

While the topology force represented by attractive and repulsive forces so it can be expressed by the following:

$$\overrightarrow{F_T} = \alpha_A \sum \overrightarrow{f_A(\iota,J)} + \alpha_R \sum \overrightarrow{f_R(\iota,J)} \quad (4)$$

α_A and α_R are coefficient that weight the attractive and repulsive forces, and $0 \le \alpha_A, \alpha_R \le 1$and $\alpha_A + \alpha_{R=}1$.

Finally the resulted force F_{RS} is express by the following:

$$\overrightarrow{F_{RS}} = \beta_G \overrightarrow{F_G} + \beta_T \overrightarrow{F_T} \quad (5)$$

Paper [3] proposed three algorithms for self-adaptive collective motion as following:

(1) Without obstacles or leaders: the predefined path P is Known for all the robots in the swarm, so every robot is responsible to maintain its position an velocity using guidance force $\overrightarrow{F_G}$ eq. (1) and topology force (repulsive force and attractive force) $\overrightarrow{F_T}$ eq. (4), while the tuning of the coefficients ratio α_A/α_R and β_G/β_T are the main steering element to keep swarm in motion as desired and reserve the topology.

(2) Without obstacles with leaders: in this case only the leader robot knows about predefined path P and the other robots just follow its motion in a collective manner, by adapting hierarchical organization to handle the limitation of directly connection with the leader due to communication range requirements. By associate each robot (i) with hierarchal index l_i depending on its closeness to the leader robot whom assigned with index=0, and the leader motion is govern as case (1) above and equations (1-5). The followers rely on the attraction force and repulsive force generated from their neighbor with lower index than them and closer to the leader. The resulted force resented in eq. (6).

$$\overrightarrow{F_{RT}} = \overrightarrow{F_T} = \alpha_A \sum_{j \in N(i), l_{j<l_i}} \overrightarrow{f_A(\iota,J)} + \alpha_R \sum_{j \in N(i)} \overrightarrow{f_R(\iota,J)} \quad (6)$$

(3) With obstacles with and without leader:

- firstly swarm with leader:–
  - If a follower robot detect an obstacle then its will be resolved to two force , as shown if figure (1*), finally the resultant force will be $\overrightarrow{F_{RS}} = \overrightarrow{F_{RS}}1$

  - If the leader detect an obstacle then it will lead the follower to avoid this obstacle by moving along boundary of that obstacle following left or right hand rule as shown in figure (2).

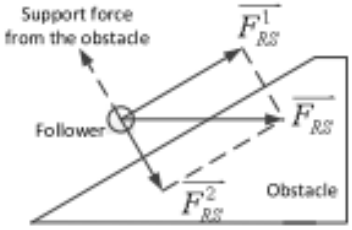


Figure 1: obstacle avoidance in follower case

- Secondly swarm without leader: if the swarm detect an obstacle the robots will elect a temporary leader to lead them following the previous approach swarm with leader. After obstacle avoidance swarm return to its normal motion without leader.

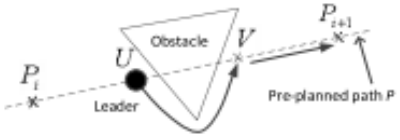


Figure 2: obstacle avoidance in leader case

This paper explores the swarm collective motion through three aspects in order to analysis the behavior of collective motion algorithm, and these aspects as follow:

- Swarm collective motion algorithm to move N number of robots through predefined path and predefined boundaries (i.e. all robots need to be close to each other to maintain proper communication. We work using linear and sine wave path.
- Swarm shape formation using collective [3], in this paper we chose the circle shape. First phase in our work is to make a robots to form a circular shape using same collective motion algorithm but the destination this time the perimeter of the circle and its center is the path coordinates. In this algorithm the constraint is consider to be the distance between the robots in order to insure uniform distribution along the circle perimeter, as shown in figure 3.

- Swarm collective motion algorithm to shape the robots in circle and move them in predefined path while the circle shape reserved. All the above algorithms use the random generation of the robots location around the starting point of the predefined path (source), as shown in figure 5.

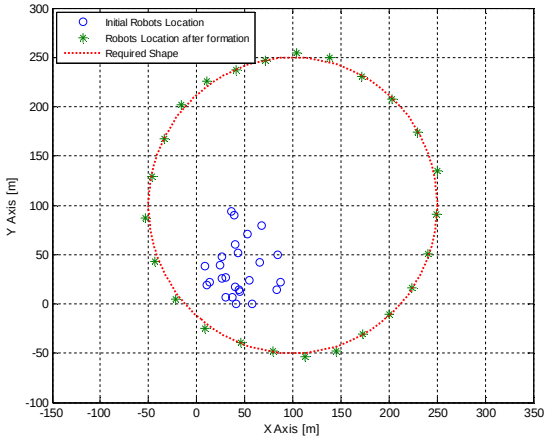


Figure 3: The figure shows the initial position of the robots and the final shape distribution (the center of the circle is (x=100, y=100) with radius =

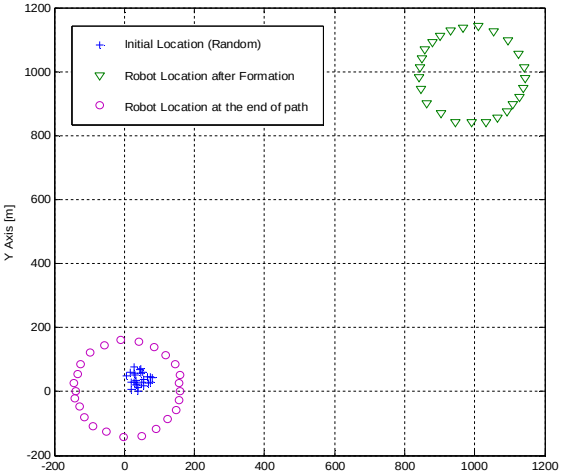


Figure 4: The figure shows the initial position of the robots and their movement from the initial position to the final along the path.

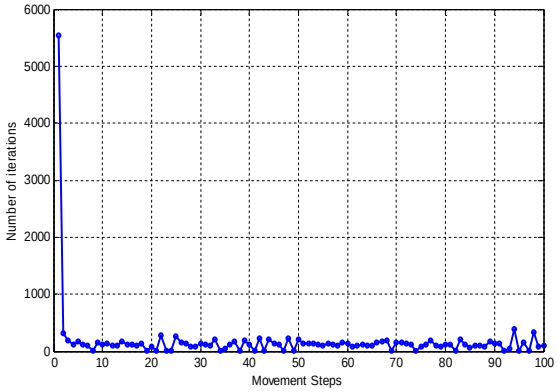


Figure 5:. The figure shows number of iterations for the algorithm to shape the robot to the circle shape and then move the robots to their final destination along the path, number of robot is 25.

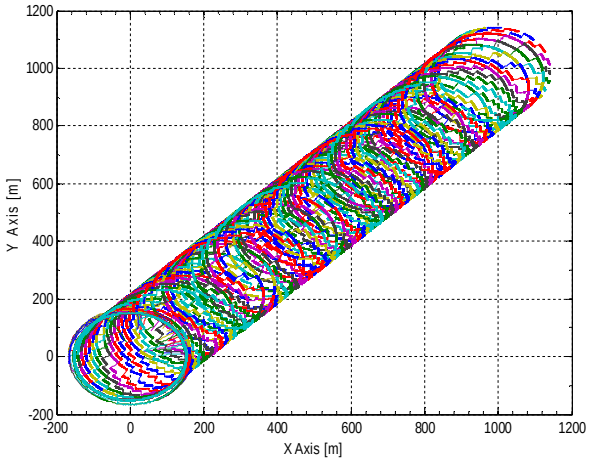
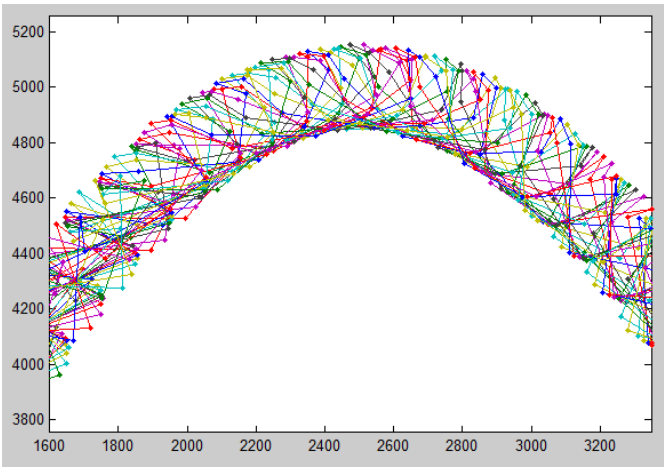


Figure 6: The figure shows history of robots movement during the iterations from source to destination, number of robot is 25.



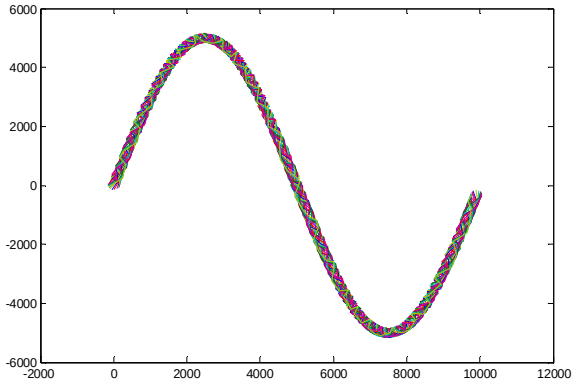


Figure.7: The figure shows history of robots movement on sinewave path

5.CONCLUSION

The use of collective motion algorithm to move set of robot shows good performance in 2-D space. This algorithm could be used to move the robots within specified boundaries and defined shape, this was achieved by manipulating constrains within the algorithm.

In future, additional work needed to adopt the optimization of the time and cost to move the robots along

the path. Although it is expected to have good performance for the 3-D space, the algorithm need to be modified to handle the 3-D space and different shape formation.

APPENDIX

- Appendix (1): algorithm
- Appendix (2): Matlab code

REFERENCES

[1] Ahmed, Hazem & Glasgow, Janice. (2012). "Swarm Intelligence: Concepts, Models and Applications". 10.13140/2.1.1320.2568.

[2] Dervis Karaboga, Bahriye Akay, "A survey: algorithms simulating bee swarm intelligence", Springer Science+Business Media B.V. 2009. https://doi.org/10.1007/s10462-009-9127-4

[3] H. Zhao, H. Liu, Y. Leung and X. Chu, "Self-Adaptive Collective Motion of Swarm Robots," in IEEE Transactions on Automation Science and Engineering, vol. 15, no. 4, pp. 1533-1545, Oct. 2018. https://doi.org/10.1109/TASE.2018.2840828

[4] Jeong, Donghwa & Lee, Kiju. "Dispersion and Line Formation in Artificial Swarm Intelligence", 2014.

[5] A. Ronzhin, I. Vatamaniuk and N. Pavluk, "Automatic control of robotic swarm during convex shape generation," 2016 International Conference and Exposition on Electrical and Power Engineering (EPE), Iasi, 2016, pp. 675-680. https://doi.org/10.1109/ICEPE.2016.7781424

[6] G. Nagy and R. Vaughan, "Self-Organization of a Robot Swarm into Concentric Shapes," 2017 14th Conference on Computer and Robot Vision (CRV), Edmonton, AB, 2017, pp. 155-160.

[7] Morin, Alexandre & Caussin, Jean-Baptiste & Eloy, Christophe & Bartolo, Denis. (2015). Collective Motion with Anticipation: Flocking, Spinning, and Swarming. Physical review. E, Statistical, nonlinear, and soft matter physics. 91. 10.1103/PhysRevE.91.012134.

[8] T. Nguyen, H. Nguyen, E. Debie, K. Kasmarik, M. Garratt and H. Abbass, "Swarm Q-Leaming With Knowledge Sharing Within Environments for Formation Control," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018, pp. 1-8.

[9] Andrew J. King & G. Cowlishaw (2009) Leaders, followers, and group decision-making, Communicative & Integrative Biology, 2:2, 147-150, DOI: 10.4161/cib.7562

[10] H. Zhao, H. Liu, Y. Leung and X. Chu, "Self-Adaptive Collective Motion of Swarm Robots," in IEEE Transactions on Automation Science and Engineering, vol. 15, no. 4, pp. 1533-1545, Oct. 2018. https://doi.org/10.1109/TASE.2018.2840828

[11] Dongdong Xu, Xingnan Zhang, Zhangqing Zhu, Chunlin Chen, and Pei Yang, "Behavior-Based Formation Control of Swarm Robots," Mathematical Problems in Engineering, vol. 2014, Article ID 205759, 13 pages, 2014. https://doi.org/10.1155/2014/205759

[12] T. Balch and M. Hybinette, "Behavior-based coordination of large-scale robot formations," in Proceedings of the 4th IEEE International Conference on MultiAgent Systems, pp. 363–364, 2000.

[13] A. Sadowska, H. Huijberts, D. Kostić, N. van de Wouw and H. Nijmeijer, "Formation control of unicycle robots using the virtual structure approach," 2011 15th International Conference on Advanced Robotics (ICAR), Tallinn, 2011, pp. 365-370. https://doi.org/10.1109/ICAR.2011.6088583

APPENDIX (1)

Initialization of:

1. Predefined path
2. Vmax
3. TH1-5, and Δ
4. Random generation of the robot location in the starting envelope
5. Shape constrains by:
   - Rc=300 #communication range
   - Rd=0.5*Rc # desired range
   - M=10 #Constant
   - N= 25 #number of robot
   - C_C=Rd half diameter of the circular shape
   - Robot_Seperation_distance=2*pi*C_C/(N-1);

LOOP (while)

- Calculate the guidance force for robots (between each robot and the starting point on the path.

$$\vec{F}_G = \overrightarrow{\sqrt{(x_{path} - x_i)^2 + (y_{path} - y_i)^2}}$$

- Self-Adaptive mechanism
  - Calculate the distance between the robots and Average distance and the standard deviation.

$$if\ Avr(i) - R_d > TH1\,, increase\ ration \frac{\alpha_A}{\alpha_R} by\ \Delta$$

$$if\ R_d - Avr(i) > TH2\,, \quad decrease\ ration\ \frac{\alpha_A}{\alpha_R} by\ \Delta$$

$$if\ SD(i) \le TH3\ \&\ R_c - Avr(i) > TH4\,, \quad increase\ ration\ \frac{\beta_G}{\beta_T} by\ \Delta$$

$$if\ SD(i) > TH5\,a\,, decrease\ ration\ \frac{\beta_G}{\beta_T} by\ \Delta$$

- Calculate the repulsive and the attractive forces between the robots

$$f_A(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$$f_R(i,j) = \begin{cases} \frac{M}{D_{ij}^2} - \frac{M}{R_d^2} & if\ 0 < D_{ij} < R_d \\ 0 & D_{ij} < R_d \end{cases}$$

- Calculate the Topology force,

$$F_T = \alpha_A \sum f_A + \alpha_R \sum f_R$$

- Calculate the resultants force for each robot

$$F_{RS} = \beta_G F_G + \beta_T F_T$$

- Calculate the velocity of each robot,

$$v_i = arctan|F_{RS}| * 2/\pi * V_{max}$$

- Update the location of each robot x_i, y_i

END LOOP

APPENDIX (2) MATLAB CODE

```
Rc=300; % communication range
Rd=0.5*Rc; % desired range
M=10;   %Constant
N= 25 ; %number of robot
C_C=Rd;

Robot_Seperation=2*pi*C_C/(N-1);

TH11=.15*Robot_Seperation;
TH22=.15*Robot_Seperation;
TH33=.15*Robot_Seperation;
TH44=.15*Robot_Seperation;
TH55=.15*Robot_Seperation;

%maximium and minimum velocity
Vmin=0;
Vmax=5;


S=[0,0]; % initial location

T=[1000,1000]; %final location

Path=[0:10:1000;  0:10:1000];
Path=Path';

Result_Force=zeros(N,2);
%Guaidance force
F_G=zeros(N,2);

%Initilization Random generation
Robot_Location_C=round(rand(N,2)*100);
Check1=0;
hh=1;

for KK=1:100

    Check1=0
    MM=1;
    while (Check1==0)

%////////////////////////////////////
////////////////////////////////////
////////////////////////////////
        %    Start : Calculate gudance
force for robots

%////////////////////////////////////
////////////////////////////////////
////////////////////////////////
        for i=1:N


Distance_M_CE(i)=(sqrt((Path(KK,1)-
Robot_Location_C(i,1))^2+(Path(KK,2)-
Robot_Location_C(i,2))^2));
            %calculate the Guidance
force
            Force_G(i,:)=(Path(KK,:)-
Robot_Location_C(i,:))*(Distance_M_CE(i)
-C_C);
        end


%////////////////////////////////////
////////////////////////////////////
////////////////////////////////
        %    END : Calculate gudance
force for robots

%////////////////////////////////////
////////////////////////////////////
////////////////////////////////


%////////////////////////////////////
////////////////////////////////////
////////////////////////////////
        %    Start : Calculate  distance
between robot and the rebulsive and
        %       attracrive Forces

%////////////////////////////////////
////////////////////////////////////
////////////////////////////////


        for i=1:N-1


Distance_M(i)=(sqrt((Robot_Location_C(i,
1)-
Robot_Location_C(i+1,1))^2+(Robot_Locati
on_C(i,2)-Robot_Location_C(i+1,2))^2));
            %calculate the attractive
and repulsive force
            %
            if (Distance_M(i)>
Robot_Seperation )

F_attr(i,:)=[(Robot_Location_C(i,1)-
```

```
Robot_Location_C(i+1,1))
(Robot_Location_C(i,2)-
Robot_Location_C(i+1,2))];
                F_repu(i,:)=[0 0];
            else if (Distance_M(i)==
Robot_Seperation)
                    F_attr(i,:)=[0 0];
                    F_repu(i,:)=[0 0] ;
                else
                    F_attr(i,:)=[0 0];
                    F_repu(i,:)=-
(M/(Distance_M(i)^2)-
M/(Robot_Seperation^2))/sqrt((Robot_Loca
tion_C(i,1)-
Robot_Location_C(i+1,1))^2+(Robot_Locati
on_C(i,2)-
Robot_Location_C(i+1,2))^2).*[(Robot_Loc
ation_C(i,1)-Robot_Location_C(i+1,1))
(Robot_Location_C(i,2)-
Robot_Location_C(i+1,2))];
                end
            end

        end
        %calculate each distance between
robots

Distance_M(N)=(sqrt((Robot_Location_C(N,
1)-
Robot_Location_C(1,1))^2+(Robot_Location
_C(N,2)-Robot_Location_C(1,2))^2));
        %calculate the attractive and
repulsive force between robots
        %
        if (Distance_M(N)>
Robot_Seperation )

F_attr(N,:)=[(Robot_Location_C(N,1)-
Robot_Location_C(1,1))
(Robot_Location_C(N,2)-
Robot_Location_C(1,2))];
            F_repu(N,:)=[0 0];
        else if (Distance_M(N)==
Robot_Seperation)
                F_attr(N,:)=[0 0];
                F_repu(N,:)=[0 0] ;
            else
                F_attr(N,:)=[0 0];
                F_repu(N,:)=-
(M/(Distance_M(N)^2)-
M/(Robot_Seperation^2))/sqrt((Robot_Loca
tion_C(N,1)-
Robot_Location_C(1,1))^2+(Robot_Location
_C(N,2)-
Robot_Location_C(1,2))^2).*[(Robot_Locat
ion_C(N,1)-Robot_Location_C(1,1))
(Robot_Location_C(N,2)-
Robot_Location_C(1,2))];
            end
        end

%////////////////////////////////////
////////////////////////////////////
////////////////////////////
        %End : calculate each distance
between robot and the rebulsive and
        %       attracrive Forces

%////////////////////////////////////
////////////////////////////////////
////////////////////////////


%////////////////////////////////////
////////////////////////////////////
////////////////////////////
        %     Start:  Adaptive code to
update the positive factors based on the
Avarage
        %             and Standards
deviation

%////////////////////////////////////
////////////////////////////////////
////////////////////////////
        %calaculate Avarage and
Standards deviation
        Avr= mean(Distance_M,2);
        SD= std(Distance_M,1,2);

        for i=1:N
            %
            if (Avr-Robot_Seperation >
TH11)
                Alpha_A=.9;
                Alpha_R=.1;
            else if (Robot_Seperation-
Avr > TH22)
                    Alpha_A=.1;
                    Alpha_R=.9;
                else
                    Alpha_A=.5;
                    Alpha_R=.5;
                end
            end


Force_T(i,:)=Alpha_A*[F_attr(i,1)
F_attr(i,2)]+Alpha_R*[F_repu(i,1)
F_repu(i,2)];
        end

        for i =1: N

            if (SD < TH33 &&
Robot_Seperation-Avr> TH44)
                Beta_G=.9;
                Beta_T=.1;
```

```
            else if (SD > TH55)
                    Beta_G=.1;
                    Beta_T=.9;
                else
                    Beta_G=.5;
                    Beta_T=.5;
                end
            end


Result_Force(i,:)=Beta_G.*Force_G(i,:)+B
eta_T.*Force_T(i,:);

        end

%////////////////////////////////////
////////////////////////////////////
//////////////////////////////
        %     End :  Adaptive code to
update the positive factors based on the
Avarage
        %             and Standards
deviation

%////////////////////////////////////
////////////////////////////////////
//////////////////////////////

        Avr_C= mean(Distance_M_CE,2);
        SD_C= std(Distance_M_CE,1,2);

        for i=1:N
            for j=1:N
                if i~=j

Distance_M_TestR(i,j)=(sqrt((Robot_Locat
ion_C(i,1)-
Robot_Location_C(j,1))^2+(Robot_Location
_C(i,2)-Robot_Location_C(j,2))^2));
                else

Distance_M_TestR(i,j)=1000;
                end
            end
        end

Minimum_Seperation=min(Distance_M_TestR)
;

Avr_SEB=mean(Minimum_Seperation);

        if (SD < TH33 &&
abs(Robot_Seperation-Avr_SEB) <  .5*
TH44 && SD_C < TH33 && abs(Avr_C-C_C) <
TH33)
            Check1=1;
        else
            Check1=0;
        end
```

```
%////////////////////////////////////
////////////////////////////////////
//////////////////////////////
        %    START  :  Calculate the
Final Force on each robot and Update the
        %    robots location

%////////////////////////////////////
////////////////////////////////////
//////////////////////////////

        for i=1:N

TTT(i)=sqrt(Result_Force(i,1)^2+Result_F
orce(i,2)^2);

V_i(i)=atan(TTT(i))*2/pi*Vmax;

Result_Force_F(i,1)=Result_Force(i,1)/TT
T(i)*V_i(i);

Result_Force_F(i,2)=Result_Force(i,2)/TT
T(i)*V_i(i);
        end


Robot_Location_C=Robot_Location_C+Result
_Force_F;



Robot_Location_CC(hh,:,:)=Robot_Location
_C;
        refresh;
        hh=hh+1;


        MM=MM+1;


////////////////////////////////////
////////////////////////////////////
//////////////////////////////
        %    END  :  Calculate the Final
Force on each robot and Update the
        %    robots location

%////////////////////////////////////
////////////////////////////////////
//////////////////////////////

    end
    Number_Iteration(KK)=MM

end
```