Multi-agent Polygon Formation using Reinforcement Learning

B. K. Swathi Prasad[1], Aditya G. Manjunath[2] and Hariharan Ramasangu[3]
[1]*Department of Electrical Engineering, M. S. Ramaiah University of Applied Sciences, Peenya, Bangalore, India*
[2]*Department of Computer Science and Engineering, M. S. Ramaiah University of Applied Sciences, Peenya, Bangalore, India*
[3]*Department of Electronics and Communication Engineering, M. S. Ramaiah University of Applied Sciences, Peenya, Bangalore, India*

Keywords: Formation, Pattern, Q-learning, Algorithm, Episode.

Abstract: This work provides details a simulation experiment and analysis of Q-learning applied to multi-agent systems. Six agents interact within the environment to form hexagon, square and triangle, by reaching their specific goal states. In the proposed approach, the agents form a hexagon and the maximum dimension of this pattern is be reduced to form patterns with smaller dimensions. A decentralised approach of controlling the agents via Q-Learning was adopted which reduced complexity. The agents will be able to either move forward, backward and sideways based on the decision taken. Finally, the Q-Learning action-reward system was designed such that the agents could exploit the system which meant that they would earn high rewards for correct actions and negative rewards so the opposite.

1 INTRODUCTION

With ever increasing applications of Multi-Agent Systems (MAS), a transferable learning method is a necessity so as to increase efficiency in the duration of adoption of such systems into a particular environment. These specifically include swarm-robot systems for surveillance, agriculture harvesting and rescue operations. Multi-Agent formation control configurations include centralized and decentralized pattern formations. The former entails no interaction among agents, whereas the opposite applies to the latter which utilizes all agents in the learning process. This part of the work focuses on decentralized reinforcement learning for Multi-Agent pattern formation. Control algorithms are adopted to perform pattern of agents, thereby achieving formation.

Popular control algorithms adopted for attaining desired geometric pattern are decentralized control algorithm (Cheng and Savkin, 2011), synchronization control (I. Sanhoury and Husain, 2012), predictive control (A. Guillet and Martinet, 2014) and Neural Network Controller and finite time controller (C. Zhang and Pan, 2014). The geometric pattern includes triangle (J. Desai and Kumar, 2001), rectangle (J. Desai and Kumar, 2001) (Cheng and Savkin, 2011) (I. Sanhoury and Husain, 2012), ellipse (A. Guillet and Martinet, 2014) (I. Sanhoury and Husain, 2012). However it is necessary to track the leaders pose (position and direction angle) while achieving the desired geometric pattern (C. Zhang and Pan, 2014) (Busoniu et al., 2006) (Gifford and Agah, 2007) (J. Alonso-Mora and Beardsley, 2011) (Ren, 2015).

The problem in formation control for a group of agents is dynamic assignment of geometric pattern. Many formation control strategies have been proposed - leaderfollower, behavioural and virtual structure/virtual leader approach (Ren, 2015) (Karimoddini et al., 2014) (Dong et al., 2015) (Rego et al., 2014) for preserving formation among agents. The control algorithms developed for pattern formation (Cheng and Savkin, 2011) (B. Dafflon and Koukam, 2013) (Ren, 2015) does not account for decentralized control configuration. Hence any pattern cannot be formed, where only few formations can be achieved.

Decentralized controller (Smith et al., 2006) (Duran and Gazi, 2010) (Krick et al., 2009) was developed to make agents form a desired geometric pattern. The pattern formations were achieved using agents' ID compared with coordinated variable (Cheng and Savkin, 2011), maintaining relative angle between the agents' position (Rezaee and Abdollahi, 2015), by adjustment of distance (Smith et al., 2006) and visualising each robot through the sensor and minimizing

Prasad B., Manjunath A. and Ramasangu H.
Multi-agent Polygon Formation using Reinforcement Learning.
DOI: 10.5220/0006187001590165
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 159-165
ISBN: 978-989-758-219-6

the control signal of actual sensor value with desired value (Krick et al., 2009). Also these patterns are restricted to only directed graph. Pattern formations is of two forms: artistic formation (J. Alonso-Mora and Beardsley, 2011) and precise formation (Gifford and Agah, 2007). These formations are required when the placement of sensors in the unstructured environment becomes inadequate. Thus learning is required such that agents can adapt and accomplish the task in the unstructured environment where high precision is required. Learning is evaluated for manipulator link control (Busoniu et al., 2006) using fuzzy value iteration method. However this method does not lead to any pattern formation as it focuses only on the motion control.

The focus of this work is pattern formation for a dynamic multi-agent system using decentralized machine learning algorithm. In this paper, 6 agents form pattern on certain quantity of interest using reinforcement learning. The learning process utilizes agents initial position scattered in the defined space. The learning curve utilizes the policy for action to be taken to move to next step and gets highest reward for reaching correct position. The agents are penalized for moving away from the desired position. The paper is structured as follows: Section 2 delves into the learning process. Results are discussed in Section 3. Section 4 describes the conclusion by giving the glimpse of the work carried out.

2 DECENTRALIZED LEARNING FOR MULTI-AGENTS

In this work, Q learning is adopted where each agent individually learns through the set of actions, $'A'$ for the given environmental state, $'S'$. In the proposed work, as the position of each agent is defined in 2D, learning problem is defined to operate parallel in both x and y coordinates. With this configuration, the agent can explore the state space independently. The proposed approach contributes to attain polygonal formation.

2.1 Proposed Structure for Learning

The decentralized structure utilizes Q learning for training agents to reach its coordinates. Policy is based on exploitation, where an action, $'Act'$ is performed using maximum Q value and is given in Eq.1.

$$Act = argmax_{a \epsilon A}\{Q(States - Space\_ID, a)\} \quad (1)$$

where $A = [-1, 1]$

Initially first agent starts to explore the states-space from $States - Space\_ID : 1$ till it reaches the specified target and lock, 0. Here lock indicates whether the agent should continue with the learning process or not. The current state of the agent is updated based on the specified condition as in line 19 of Algorithm 1. Once the state of the agent gets updated, the computation of Q matrix is repeated using line 34 in Algorithm 1, to pre-compute the policy adopted until the specified target is reached. For the next agent, the target position of previous agent is taken as the initial state of the states-space and learns through the Q-learning until it reaches the specified target.

2.2 Algorithm to Compute Reward of the Agent

This section focuses on the computation of reward of the agent for the defined state space. An agent is given maximum reward if it performs some action and is penalized if the agent performs action even after reaching the target. In Section 2.2.1 and Section 2.2.2, the assignment of rewards for x-coordinate and y-coordinate are derived from the specific target assigned to each agent respectively.

2.2.1 Algorithm for Computing Reward of x-coordinate of Agent

The next state of each agent is computed and updated based on the action taken from the optimum Q value. The next state of each agent ID is updated till it reaches the goal state of x-coordinate and lies within the goal states space of $[-2, 2]$. The agent gets maximum reward when it performs correct action, gets penalty when the agent reaches goal state of $-1, 0, 1, 2, 1, 0$ and also if neither of the case then the agent is facilitated with lesser negative reward. The complete structure of algorithm for computing next state of each agent ID is described in Algorithm 2.

2.2.2 Algorithm for Computing Reward of y-coordinate of Agent

The next state of each agent is computed and updated based on the action taken from the optimum Q value. The next state of each agent ID is updated till it reaches the goal state of y-coordinate and lies within the goal states space of $[-1, 1]$. The agent gets maximum reward when the agent performs certain action, gets penalty when the agent reaches goal state of $0, 1, 1, 0, -1, -1$ and also if neither of the case then the agent is facilitated with lesser negative reward. The complete structure of algorithm for comp-

Algorithm 1: Proposed Structure for Learning.

Require: $States-Space, Action, Q$
Require: Learning Rate and Discount Factor
$States-Space = -2:3$
$Action$= [move down (-1), move up (1)]
Learning Rate, $alpha = 0.5$
Discount Rate, $gamma = 0.5$
Q = zeros(length (States-Space), length (Action))
$States-Space\_ID$ = Index of $States-Space$
N = Total no. of agents
M = Total no. of iterations
x = Current State
y = Next State
$Next\,State\_ID$ = Index of agent for its next position in the $States-Space$
$lock$ = To indicate whether to restrict the movement or not
$Target$ = Goal position of the agent
repeat
  for $i \leftarrow 1, N$ **do**
    $States-Space\_ID = 1$
    repeat
      for $k \leftarrow 1, M$ **do**
        $Action\_ID = find(Action == max(Q(States-Space\_ID,:))$
        $x(i) = States-Space(States-Space\_ID)$
        Comment: Based on current action and i, next state y is computed
        if $x \leq max(States-Space) \&\& x \geq min(States-Space)$ **then**
          $y(i) = x(i) + Action(Action\_ID)$
        else
          $y(i) = x(i)$
        end if
        Comment: Based on Current State and Action, Assign Reward
        if $y == Target$ **then**
          $lock = 1$
        else
          $lock = 0$
        end if
        if $y == -3$ **then**
          $y = -2$
        end if
        $Next\ State\_ID = find(States-Space == y)$
        Update Q using Q-Learning Rule
        $Q(States-Space\_ID, Action\_ID) = (States-Space\_ID, Action\_ID) + alpha(Reward + gamma * max(Q(Next\ State\_ID,:))) - Q(States-Space\_ID, Action\_ID))$
        $States-Space\_ID = Next\ State\_ID$
      end for
    until $y == Target$
  end for
until $lock==1$

Algorithm 2: Computation of Reward for x- coordinate of Each Agent.

Require: Current State, Action, Reward
x = Current State
u = Action
r = Reward
N = Total no. of agents
t = Target
for $i \leftarrow 1, N$ **do**
  if $i == 1$ **then**
    Target = -1
  else if $i == 2 \mid i == 6$ **then**
    Target = 0
  else if $i == 3 \mid i == 5$ **then**
    Target = 1
  else
    Target = 2
  end if
  if $x == t-1 \&\& u == 1$ **then**
    $r = 10$
  else if $x == t+1 \&\& u == -1$ **then**
    $r = 10$
  else if $x == t \&\& u == 1 \parallel u == -1$ **then**
    $r = -10$
  else
    $r = -1$
  end if
end for

uting next state of each agent ID is described in Algorithm 3.

2.2.3 Algorithm for Plotting Agents Learning Process

The Algorithm 4 briefs about the updated state of each agent for a given episode. This is required to know whether the agent has reached its goal state or not.

3 RESULTS AND DISCUSSION

This section details agents learning to reach their target along x and y coordinates, and the episodes they have undergone to achieve the same. With the next state history of each agent along x and y coordinates, the desired pattern of hexagon is achieved.

3.1 Transition Along x Coordinate to Reach Target

The agents after undergoing the phase of algorithm described in Section 2.2.1, it traverses several episo-

Algorithm 3: Computation of Reward for y-coordinate of Each Agent.

Require: Current State, Action, Reward
1: x = Current State
2: u = Action
3: r = Reward
4: N = Total no. of agents
5: t = Target
6: **for** $i \leftarrow 1, N$ **do**
7: **if** $i == 1 \mid i == 4$ **then**
8: Target = 0
9: **else if** $i == 2 \mid i == 3$ **then**
10: Target = 1
11: **else**
12: Target = -1
13: **end if**
14: **if** $x == t-1 \&\& u == 1$ **then**
15: $r = 10$
16: **else if** $x == t+1 \&\& u == -1$ **then**
17: $r = 10$
18: **else if** $x == t \&\& u == 1 \parallel u == -1$ **then**
19: $r = -10$
20: **else**
21: $r = -1$
22: **end if**
23: **end for**

Algorithm 4: Plot of Learning of agents to reach target with episodes incurred by each agent.

Require: next state history
1: P_x = Positions of x coordinate
2: P_y = Positions of y coordinate
3: R_x = next state_x
4: R_y = next state_y
5: E = Episode of Agent ID
6: $P_x = [R_x, E]$
7: $P_y = [R_y, E]$
8: N = Total no. of agents
9: **for** $i \leftarrow 1, N$ **do**
10: $s1 = find(R_x) == i$
11: $s2 = find(R_y) == i$
12: $jj1 = max(s1)$
13: $jj2 = max(s2)$
14: $plot(i, R_x(jj1, 1)$
15: $plot(i, R_y(jj2, 1)$
16: **end for**

des defined in the state space. This transition of agents from one state to the other is shown in Fig. 1. It is seen that agents reach to the goal state as indicated in Algorithm 2 of Section 2.2.1. The episodes for transition is indicated in Table 1.

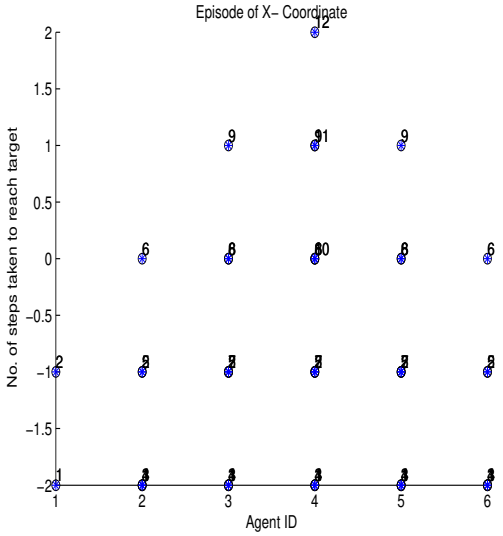

Figure 1: Transition of Agents x coordinate to reach target.

Table 1: Episode incurred by each agents x coordinate.

Agent ID	No. of Episodes Incurred
1	2
2	6
3	9
4	12
5	9
6	6

3.1.1 Learning x Coordinate to Reach Target

The agent reaches its target from the initial position of each agent in x coordinate as shown in Fig.2

3.1.2 Transition Along y Coordinate to Reach Target

The agents after undergoing the phase of algorithm described in Section 2.2.2, it traverse through several episodes defined in the state space. This transition of agents from one state to the other is shown in Fig. 3. It is seen that agents reach to the goal state as indicated in Algorithm 3 of Section 2.2.2. The episodes for transition is indicated in Table 2. The critical analysis was executed for a test data to check the variation of number of episodes incurred for different initial position and target for the same. It is observed that when the agent needs to travel from lower negative value to higher negative value or lower positive value to higher positive value, more number of episodes are taken by the agent to reach its target. However if the agent traverses from higher negative value to lower negative value and higher positive value to lower positive va-

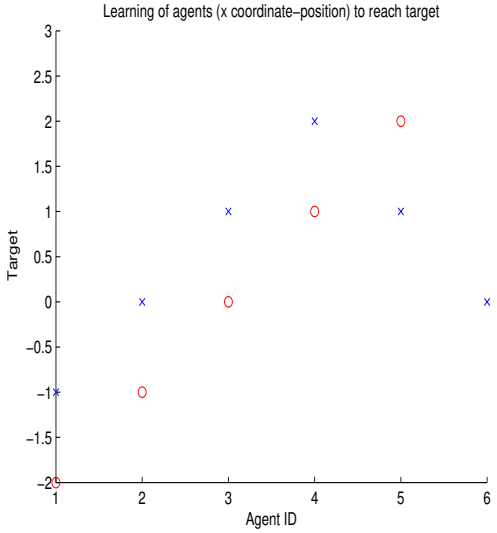

Figure 2: Learning of agents x coordinate.

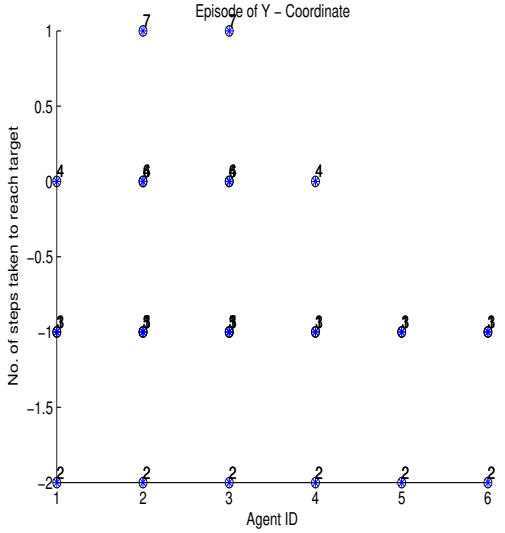

Figure 3: Transition of Agents y coordinate to reach target.

Table 2: Episode incurred by each agents x coordinate.

Agent ID	No. of Episodes Incurred
1	4
2	7
3	7
4	4
5	3
6	3

lue, agents takes lesser episodes comparatively. This analysis signifies the computational time incurred by each agent to reach its target.

3.1.3 Learning Along y Coordinate to Reach Target

The agent reaches its target from the initial position of each agent in y coordinate as shown in Fig. 4

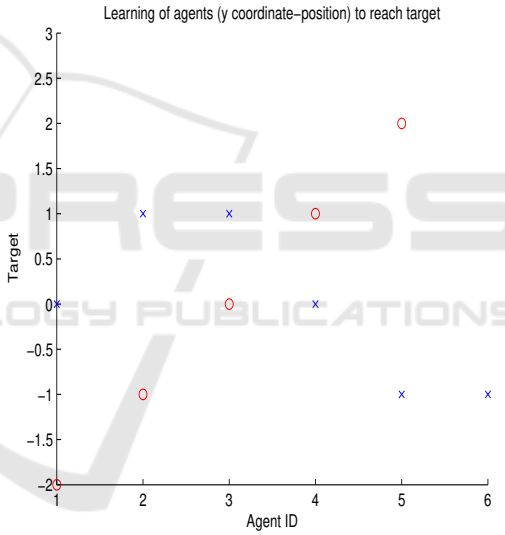

Figure 4: Learning of agents y coordinate.

3.2 Pattern Formation

Patterns from maximum dimension (6 / hexagon) to reduced polygon dimensions of the size three, four and five were achieved in this work. To suit all achievable patterns, the target assignment defined in Algorithm 2 of Section 2.2.1 and Algorithm 3 of Section 2.2.2 utilized to obtain the desired pattern. Before forming pattern, the agent utilizes initial position as the target position of previous agent position and reaches the specified target through learning. Hence the agents are connected and cooperative to form the specified pattern.

While forming the pattern, certain agents cannot sense or fail to perceive the information about the presence of other agents. Such agents are eliminated to form patterns of smaller dimensions. The elimination of certain agents for obtaining the desired pattern is shown in Table 3.

Table 3: Elimination of Agents to Obtain the Desired Pattern.

Pattern	Cases to Perform Different Patterns
Hexagon	No agents are eliminated
Pentagon	Agent 4 is eliminated
Square	Agent 1 and Agent 4 is eliminated
Triangle	Agent 3, Agent 4 and Agent 5 is eliminated

The patterns of hexagon, pentagon, square and tri-

angle are shown in Fig. 5, Fig. 6, Fig. 7 and Fig. 8 respectively. The notations for initial and desired positions are represented by $'o'$ and $'x'$ respectively. With this proposed design, patterns of various reduced dimensionality (from the maximum of that of a hexagon) can be demonstrated and not just restricted to the patterns shown in this paper.

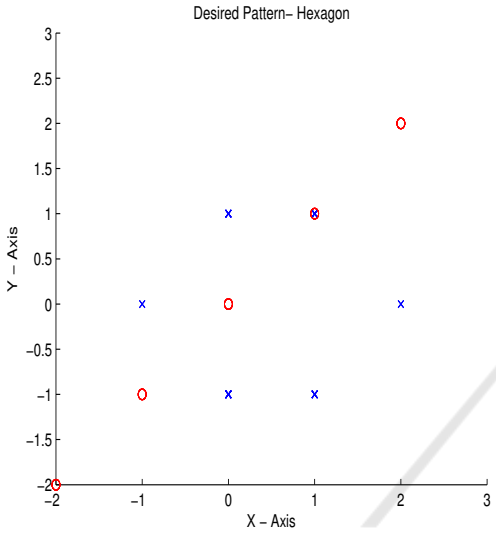

Figure 5: Desired Pattern: Hexagon.

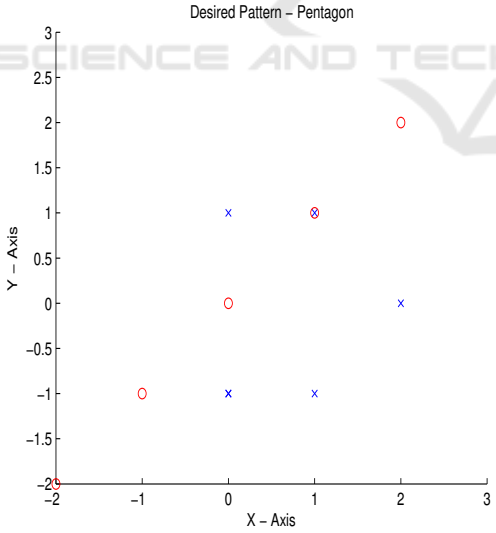

Figure 6: Desired Pattern: Pentagon.

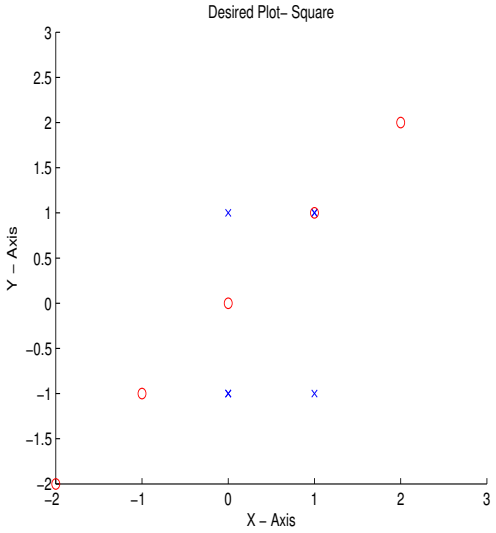

Figure 7: Desired Pattern: Square.

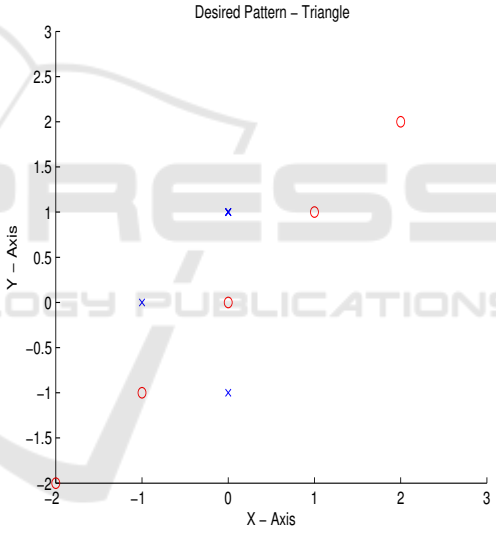

Figure 8: Desired Pattern: Triangle.

4 CONCLUSION AND FUTURE WORK

This work demonstrates a practical method for pattern formations in MAS. The action-reward system of Q-learning is ideal choice for formation of patterns in a behavioural based system (such as the one demonstrated in this work), as it allows for a rigid control on the movements through exploitation. Longer learning periods are a consequence of freedom to explore the environment, which means reducing the number of possibilities of selecting the correct action (higher re-

ward) is not beneficial for formation of patterns. The method demonstrated here is best suited for a known environment. Applications of MAS patterns are vast, and this method demonstrated in this paper is highly adaptable and user friendly to account for any patterns as desired. Proof of concept of this research is presented over the formation of polygons from maximum hexagonal dimension to minimum dimensionality of triangular. In this work, the control of agents to reach the specified target is controlled independently for both x and y position of agent. This can be advantageous in both computational efforts and time. To validate with other methods, the pattern formation was tested using neural network. The drawback is each agent should be specified with its initial position and target. States-Space search cannot be obtained by using this approach.

Future work includes combining leader-follower (Prasad et al., 2016a) (Prasad et al., 2016b) trajectory tracking with pattern formation. We would like to keep pattern selection and formation under the control of the leader. Coupling this system with leader election would also be interesting and would help counter any loss of connectivity during trajectory tracking.

REFERENCES

A. Guillet, R. Lenain, B. T. and Martinet, P. (2014). Adaptable robot formation control: Adaptive and predictive formation control of autonomous vehicles. In *IEEE Robotics and Automation Magazine*. IEEE.

B. Dafflon, F. Gechter, P. G. and Koukam, A. (2013). A layered multi-agent model for multi-configuration platoon control. pages pp.33–40. International Conference on Informatics in Control, Automation and Robotics.

Busoniu, L., Schutter, B. D., and Babuska, R. (2006). Decentralized reinforcement learning control of a robotic manipulator. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6.

C. Zhang, T. S. and Pan, Y. (2014). Neural network observer-based finite-time formation control of mobile robots. pages pp. 1–9. Mathematical Problems in Engineering.

Cheng, T. and Savkin, A. (2011). Decentralized control of multi-agent systems for swarming with a given geometric pattern. pages 61(4), pp.731–744. Computers and Mathematics with Applications.

Dong, X., Yu, B., Shi, Z., and Zhong, Y. (2015). Time-varying formation control for unmanned aerial vehicles: theories and applications. *IEEE Transactions on Control Systems Technology*, 23(1):340–348.

Duran, S. and Gazi, V. (2010). Adaptive formation control and target tracking in a class of multi-agent systems. In *Proceedings of the 2010 American Control Conference*, pages 75–80.

Gifford, C. M. and Agah, A. (2007). Precise formation of multi-robot systems. In *2007 IEEE International Conference on System of Systems Engineering*, pages 1–6.

I. Sanhoury, S. A. and Husain, A. (2012). Synchronizing multi-robots in switching between different formations tasks while tracking a line. pages pp.28–36. Communications in Computer and Information Science.

J. Alonso-Mora, A. Breitenmoser, M. R. R. S. and Beardsley, P. (2011). Multi-robot system for artistic pattern formation. pages pp. 4512–4517. Robotics and Automation (ICRA), IEEE International Conference.

J. Desai, J. O. and Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. pages 17(6), pp.905–908. IEEE Trans. Robot. Automat.

Karimoddini, A., Karimadini, M., and Lin, H. (2014). Decentralized hybrid formation control of unmanned aerial vehicles. In *2014 American Control Conference*, pages 3887–3892. IEEE.

Krick, L., Broucke, M. E., and Francis, B. A. (2009). Stabilisation of infinitesimally rigid formations of multi-robot networks. *International Journal of Control*, 82(3):423–439.

Prasad, B. K. S., Aditya, M., and Ramasangu, H. (2016a). Flocking trajectory control under faulty leader: Energy-level based election of leader. pages 3752–3757. IEEE.

Prasad, B. K. S., Aditya, M., and Ramasangu, H. (2016b). Multi-agent trajectory control under faulty leader: Energy-level based leader election under constant velocity. pages 2151–2156. IEEE.

Rego, F., Soares, J. M., Pascoal, A., Aguiar, A. P., and Jones, C. (2014). Flexible triangular formation keeping of marine robotic vehicles using range measurements 1. *IFAC Proceedings Volumes*, 47(3):5145–5150.

Ren, W. (2015). Consensus strategies for cooperative control of vehicle formations. In *Control Theory and Applications*, pages pp.505 – 512. IET.

Rezaee, H. and Abdollahi, F. (2015). Pursuit formation of double-integrator dynamics using consensus control approach. *IEEE Transactions on Industrial Electronics*, 62(7):4249–4256.

Smith, S. L., Broucke, M. E., and Francis, B. A. (2006). Stabilizing a multi-agent system to an equilateral polygon formation. In *Proceedings of the 17th international symposium on mathematical theory of networks and systems*, pages 2415–2424.