Technical Report

Authors : *Filippos Christou , Ioannis Gkouziwkas and Spyros Fontalis*

Title : ***Decentralized Multi-Agent Estimation and Formation Control***

Abstract: *The goal of this paper is to present a robust and scalable framework for the design of collective behaviors for groups of identical mobile agents. This task involves the presentation of the necessary mathematical work and assumptions, as well as the appliance of these theoretical tools in the creation of a relevant simulation. The approach is based on decentralized simultaneous control and estimation as described by* [1]*. That means that instead of creating a global control law and assuming perfect knowledge of the global performance, properties of the swarm are being estimated by every agent. Our approach is based on the mutual agent communication and individual agent estimation of these global properties, which in turn are needed as input for the function of a local control law, which runs in each agent individually. The methodology adopted for the completion of this task includes firstly, the design of a control law with desired convergence properties, with the primary assumption that each agent has perfect global knowledge, secondly, the design of an estimator that enables individual agents to make correct estimates over the global properties. Thirdly, we provide these estimates to the controller (making it, this way decentralized). Finally, we modify the controller so that we achieve the desired convergence properties, since we are using the global performance estimates and not the actual statistics. This procedure is used to control the moment statistics of the swarm which describe its location and shape, even in the presence of a changing network topology.*

INTRODUCTION

The recent advances in electronics and wireless communication has provided us with a new interesting and full of potential technological tool at our disposal. The development of low-cost, low-power, multi-functional, small in size sensor nodes, which are constituted by sensing, data processing and communicating components, inspire the idea of sensor networks. The potential of these networks is skyrocketing and the positive effect their adjustability and functionality can have on our everyday lives is surely fascinating. Some example applications include managing inventory, monitoring disaster areas and monitoring product quality.

Some design challenges that derive from the physical and hardware constraints of the nodes are the wise management of the limited power, computational capacity and memory of the nodes and also the design of scalable algorithms.

Our exploitation of the mobile agents' capabilities is based heavily on their controlled mobility. The goal of the paper is to present a global dynamics system which behaves as an "attractor". Thus, given a set of identical mobile agents, the interaction of the control laws (which are being supplied with sensor and communication input and are created to run on each agent) has a goal to achieve specific collective "emergent" global behavior.

The main design principles of this approach are its scalability, robustness and the inexistence of a global, central controller. That means that the performance of the swarm should not be affected by the deletion of already working nodes or the addition of new ones.

This approach is based heavily upon the each agent's individual sensing, mutual communication and again individual estimation of the global properties of the group. It is proven in a number of models of schooling and flocking [2] [3] [4] [5] that systems in which each agent is ignorant of the global properties of the system, there exist the capability of achieving intelligent collective behavior. The problem is though, that in these systems there are few tools to guide the design of local control laws to achieve a particularly desired collective behavior. Reactive controllers using immediate sensory and communication input from neighbors (memoryless agents) can solve only a limited number of tasks, and often require too specific design of the communication and sensing network. Therefore, our will to analyze a methodology which can be

implemented to solve an abundance of tasks has led us to a framework with changing number of agents and few requirements on the time-varying communication network (agents with memory).

In the most basic and abstract level of this framework, the desired behavior of the group is encoded in a global cost function J. Then, the agents are equipped with an estimator of the global properties of the system, which uses sensory data and information gained from other agents, and a local controller whose inputs are the estimated properties of the swarm and its goal is to minimize the global cost function. The use of estimated input for our controller compel us to modify our controller accordingly, so that we achieve the desired convergence properties.

The immediate applications of this estimate-and-control approach are both motion coordination tasks and mobile sensing tasks. In this paper, we focus on a formation control example, where the global cost is only affected by the individual positions of the agents.

PROBLEM FORMULATION

The system consists of a collection of *n* identical mobile agents, each implementing the same control system with no identifying tags or ID numbers. Agents may be added or deleted from the system any time without changing its functionality. The physical state of each agent *i* is expressed as the vector p_i, its control the vector u_i, and its state evolves according to the dynamics

$$\dot{p_i} = F(p_i, u_i). \tag{1}$$

We write the combined vector of positions of the agents as $p = [p_1^T \dots p_n^T]^T \in \mathbb{R}^{mn}$. Then we represent the collection of all possible swarm configurations as the topological coproduct $\mathfrak{B} \equiv \coprod_{n=1}^{\infty} \mathbb{R}^{mn}$. A finite number of geometric moments can be used to encode an abstraction of a robot formation [6]. This abstraction will allow the easy and simple physical control of the swarm from the directions of the human handler. If the position of agent *i* in a plane is denoted $p_i = \left[p_{ix}, p_{iy}\right]^T$, then a formulation of *n* robots can be abstracted to an *l*-vector of moments of the form

$$f(p) = \frac{1}{n}\sum_{i=1}^{n}\left[p_{ix}p_{iy}p_{ix}^2p_{iy}^2p_{ix}p_{iy}p_{ix}^3 \dots\right]^T \tag{2}$$

,where $p = [p_1^T \dots p_n^T]^T$.

The application we explore in this paper involves formations defined by first- and second-order moments. The *m* first order moments specify the center of mass of the swarm. From the *m(m+1)/2* second-order moments we can derive *m(m-1)/2* variables describing the orientation of orthogonal principal axes of inertia of the swarm and m shape variables summarizing the elongation of the swarm along the principal axes.

We describe the desired swarm configuration using a vector goal function $f: \mathfrak{B} \to \mathbb{R}^l$. With f^* being a goal vector known to each agent, the primary objective of each agent is to move itself to an equilibrium position so that the final swarm configuration *p* satisfies $f(p) = f^*$.

The total number *n* of agents in the swarm is unknown to each agent.

Each agent takes measurements of the form

$$z_i = C(p_i, p_i^{sens}), \tag{3}$$

,where p_i^{sens} represents the states of other agents that affect or contribute to the measurements. The vector z_i may measure absolute or relative positions and velocities of the agents, or environmental variables such as temperature, salinity, or locations of points of interest in the environment. The dimension of z_i may change depending on the number of neighbors in the sensing network and surely cannot be higher than the overall agents.
Therefore, each agent measures its own position and velocity and controls its position derivative, speed.
Furthermore, each agent can communicate with its neighbors; specifically, agents *i* and *j* can communicate with each other whenever $p_i \leftrightarrow p_j$. where $\leftrightarrow$ is a fixed symmetric relation on $\mathbb{R}^m$.

Crucial importance to this implementation and the most important factor that differentiates it from other methodologies is the design of our estimator.

Our dynamic average consensus estimator is based on the work of Olfati-Saber and Murray [7], who studied the convergence of the differential equation

$$\dot{x} = -Lx \tag{4}$$

,where x is a decision variable, which indicates our estimator, and L is the Laplacian of a communicator graph of agents. They showed that each agent's decision variable x_i converges to a common value, which is the average of the initial values $x_i(0)$. That is, if the communication graph is strongly connected and balanced, which means that for each node in the graph, the in-degree and out-degree are equal. Very useful is the fact that average consensus is reached even with switching network topologies and uniform communication delays bounded by a function of the largest eigenvalue of L.

Many functions of sensor inputs can be estimated using average consensus estimators, as we can first apply a transformation to the sensor inputs, pass them through the averaging process, and then apply a transformation to the output to obtain the desired result.

This average consensus problem is referred to as static, since it only involves the initial data $x(0)$. An estimator capable of tracking the average of changing inputs, a *high pass consensus estimator* was proposed in [8], while dynamic consensus estimators with improved noise rejection and steady-state performance in the face of addition or deletion of agents is introduced in [9].
The type of dynamic average consensus estimator we apply in this paper is a proportional-integral (PI) consensus estimator.

Suppose each agent implements a proportional-integral dynamic consensus estimator of the form

$$\begin{aligned}\dot{x}_i(t) = -\gamma x_i(t) &- \sum_{j\neq i} a_{ij}(t)\big[x_i(t) - x_j(t)\big] \\ &+ \sum_{j\neq i} b_{ji}(t)\big[w_i(t) - w_j(t)\big] \\ &+ \gamma r_i(t)\end{aligned} \tag{5}$$

$$\dot{w}_i(t) = -\sum_{j\neq i} b_{ij}(t)\left[x_i(t) - x_j(t)\right] \tag{6}$$

where $r_i(t) \in \mathbb{R}$ is the input, $x_i(t) \in \mathbb{R}$ is the estimator output, $w_i(t)$ is the internal estimator state, $\gamma > 0$ is a global estimator parameter, and $a_{ij}(t)$ and $b_{ij}(t)$ are piecewise-continuous time-varying estimator gains. The constraint $a_{ij}(t) = a_{ji}(t) = b_{ij}(t) = b_{ji}(t) = 0$ is imposed if agents *i* and *j* cannot communicate with each other at time *t*. We may write the collection of these *n* estimators in vector form as

$$\begin{bmatrix}\dot{x}(t)\\ \dot{w}(t)\end{bmatrix} = \begin{bmatrix}-\gamma I - L_p(t) & L_I^T(t)\\ -L_I(t) & 0\end{bmatrix}\begin{bmatrix}x(t)\\ w(t)\end{bmatrix} + \begin{bmatrix}\gamma I\\ 0\end{bmatrix} r(t), \tag{7}$$

,where $L_p(t)$ is the proportional Laplacian (constructed from the weights a_{ij}) and $L_I(t)$ is the integral Laplacian (constructed from the weights b_{ij}).

This estimator has no direct feedthrough from its input r_i to its output y_i, and thus it may provide better filtering of high frequency noise. A "forgetting" factor γ in this PI filter results in a stable filter from communication noise to errors relative to the solution manifold.
For the PI consensus estimator, it is proven that assuming L_P, L_I are constant Laplacians, $\varepsilon \in \mathbb{R}$ is such that $L_I \in \mathcal{L}^T, L_P \in \mathcal{L}^E$, rank $L_I = n - 1$ and the estimator parameter $\gamma > 0$ is chosen such that $\gamma + \varepsilon > 0$, then for any constant input $u \in \mathbb{R}^n$ and any initial states $x(t_o)$, $w(t_0) \in \mathbb{R}^n$, the trajectories of the system

$$\begin{bmatrix}\dot{x}(t)\\ \dot{w}(t)\end{bmatrix} = \begin{bmatrix}-\gamma I - L_p(t) & L_I^T(t)\\ -L_I(t) & 0\end{bmatrix}\begin{bmatrix}x(t)\\ w(t)\end{bmatrix} + \begin{bmatrix}\gamma I\\ 0\end{bmatrix} u \tag{8}$$

are such that $x(t)$ and $w(t)$ converge to constant vectors and the error vector $e_x(t) \to 0$ approaches zero as $t \to \infty$.

Our approach is based on following estimates of the gradient ∇J of the potential function $J: \mathcal{B} \in \mathbb{R}$ given by

$$J = [f(p) - f^*]^T \Gamma [f(p) - f^*] \tag{9}$$

,where $\Gamma \in \mathbb{R}^{l\times l}$ is a suitably chosen symmetric positive-definite global gain matrix. The

$$Crit(J) \equiv \{p \in \mathfrak{B}: \nabla J(p) = 0\} \tag{10}$$

is defined to denote the set of critical points of J, and such points are classified as "good" critical points where $f(p) = f^*$ (these are the global minima of *J*) and "bad" critical points where $f(p) \neq f^*$. One of our major goals is for the swarm to not get stuck at such "bad" critical points.

We are interested in dynamic local state feedback controllers of the form

$$u_i = K(p_i, z_i, \eta_i, y_i, S_i) \quad (11)$$
$$s_i = G(p_i, z_i, \eta_i, y_i, S_i) \quad (12)$$
$$\dot{\eta}_i = Q(p_i, z_i, \eta_i, y_i, S_i) \quad (13)$$
$$y_i = R(p_i, z_i, \eta_i, S_i) \quad (14)$$

where s_i denotes the fixed-dimension signal vector agent i sends to its neighbors, S_i denotes the variable-dimension collection of signals agent receives from its neighbors through communication channels, the vector η_i is the internal estimator state containing information about the global performance of the system, and the vector y_i represents the output of the estimator.

Implicit in the formulation are three dynamic interaction networks: a network of physical interactions, represented by $\{X_i^{phys}\}$; a network of sensing interactions, represented by $\{X_i^{sens}\}$; and a network of communication interactions, implicit in $\{S_i\}$. These networks may change over time.

The goal of our work is to design a local controller K, a signal generator G, and a state estimator Q and R to minimize a cost functional J encoding the desired behavior of the system :

$$J = \varphi\big(Y(t_f), t_f\big) + \int_{t_0}^{t_f} L(Y(\tau), u(\tau), \tau) d\tau \quad (15)$$

,where $u = [u_1^T \dots u_n^T]^T$ and Y represents the collection of all x_i, η_i and y_i.

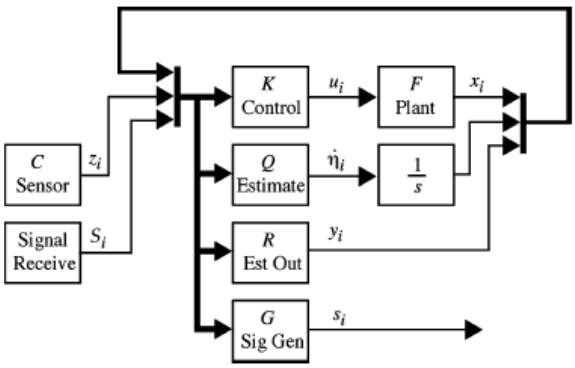

Figure 1: Block diagram for agent i.

METHODOLOGY

Each robot is modeled as a point mass with the column vector $p_i \in \mathbb{R}^m$ signifying its position. For our simulation, the number of dimensions is $m = 2$ (planar) and so $p_i = [p_{ix}, p_{iy}]^T$. The state of the robot is $x_i = [p_i^T, \dot{p_i}^T]^T$, with dynamics

$$\dot{x_i} = \begin{bmatrix} \dot{p_i} \\ u_i \end{bmatrix}, \quad (16)$$

,where $u_i \in \mathbb{R}^{2n}$ is the control force. For *n* robots, the configuration of the entire swarm is written $p = [p_1^T, \dots, p_n^T]^T \in \mathbb{R}^{2n}$.

Since our example focuses on a formation control, we need an encoding of the swarm formation. To do this, we use a finite number of geometric moments. We define a goal formation for the swarm by a set of desired first and second order inertial moments. The first moments specify the swarm's center of mass and the second moments specify its direction-dependent spread in the plane. For a set of unit-mass point robots in the plane, the five mass-normalized first and second moments expressed in a fixed global frame can be written as the vector

$$f(p) = \frac{1}{n}\sum_{i=1}^{n} \varphi(p_i) = \frac{1}{n}\sum_{i=1}^{n} [p_{ix}, p_{iy}, p_{ix}^2, p_{ix}p_{iy}, p_{iy\dots}^2]^T, \quad (17)$$

,where $\varphi(p)$ is called the moment-generating function.

Each robot measures its own position and knows f^*. To achieve a desired vector of moments f^*, the cost function (9) was defined. This function is used to drive the robots to the goal configuration submanifold. The control law implemented by robot *i* simply drives it along the negative gradient of this potential, with added damping so that the robots come to rest on the goal submanifold. The control law is

$$u_i = -B\dot{p_i} - [\Im\varphi(p_i)]^T \Gamma[f(p) - f^*] \quad (18)$$

,where $B \in \mathbb{R}^{2x2}$ is a positive-definite damping matrix and $\Im\varphi(\cdot)$ denotes the 5x2 Jacobian matrix of φ with respect to p_i. Note that the control law involves the swarm's current moments $f(p)$, a global quantity. In a decentralized implementation, each robot must replace $f(p)$ with a local estimate

obtained through communication with neighboring robots.

To obtain local estimates of the global swarm moments, each robot implements a proportional-integral (PI) average consensus estimator. Each robot continuously receives its neighbors' estimates of the swarm's moments, averages them with its own, and broadcasts its new estimate. Inputs to the averaging process are the robots' measurements of their own positions. More specifically, robot *i* broadcasts to its neighbors its estimator output u_i and its estimator internal state w_i, and calculates

$$\dot{v}_i = \gamma(\varphi(p_i) - v_i) - \sum_{j \neq i} a(p_i, p_j)[v_i - v_j] + \sum_{j \neq i} b(p_i, p_j)[w_i - w_j] \quad (19)$$

$$\dot{w}_i = -\sum_{j \neq i} b(p_i, p_j)\,[v_i - v_j], \quad (20)$$

as we expected from the Eq. (5) and (6). The same properties apply here too : $a(p_i, p_j) \geq 0$ is a proportional gain between robots *i* and *j*, $b(p_i, p_j) \geq 0$ is an integral gain between robots *i* and *j*, and $\gamma > 0$ is a global "forgetting factor". The gains $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ implement an undirected graph. When robots *i* and *j* cannot communicate, $a(p_i, p_j) = 0$ and $b(p_i, p_j) = 0$. New information enters the averaging process through the term $\gamma(\varphi(p_i))$, with controlling the rate at which new information replaces old information.

The PI average consensus estimator results in estimates v_i that track $f(p)$ closely, provided that the communication network is connected and $f(p)$ is changing slowly. Furthermore, if the inputs stop changing (*p* is constant), the estimates converge to $f(p)$ with zero steady-state error. This behavior occurs even after dropped packets, noisy messages, and robots entering or leaving the network.

Substituting the estimator output v_i for the actual moments $f(p)$ yields the decentralized control law for robot *i*

$$u_i = -B\dot{p}_i - [\Im\varphi(p_i)]^T \Gamma [v_i - f^*] \quad (21)$$

This slight modification of this control law, used along with the PI estimator, will cause the swarm to converge to the desired configuration submanifold from nearly every initial state, and all equilibria away from the goal submanifold are unstable.

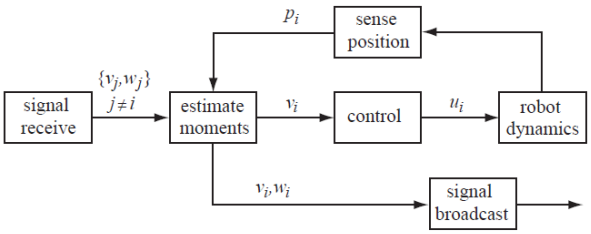

Figure 2: Block diagram for agent i

To implement the theoretical estimation and control procedure described above on a real implementation, we made three primary modifications:

Firstly, the control law was modified for kinematic robots with velocities as controls, instead of point masses with forces as controls. This change makes our model of moving agents somewhat more natural and simpler to apply. Therefore, the state of the system is simply the robot's configuration, $x_i = y_i$, and the dynamics are written $\dot{x}_i = u_i$. With these dynamics, we simplify the gradient law

$$\acute{u}_i = -[\Im\varphi(p_i)]^T \Gamma [u_i - f^*] \quad (22)$$

Kinematic robots under this control law guarantee performance similar to those of point mass robots with force inputs under the control law.

Secondly, the definition of second moments referenced to the origin, was modified to central second moments, referenced to the swarm's center of mass. In our previous formula for calculating the group's moments, the moment-generating function $\varphi(p_i)$ was defined as

$$f(p_i) = \left[p_{ix}, p_{iy}, p_{ix}^2, p_{ix}p_{iy}, p_{iy}^2 \ldots\right]^T \quad (23)$$

This choice has the disadvantage that as the swarm translates without changing shape, the second moments change. These global second moments are with respect to the origin, not the center of mass of the swarm. The result is that the behavior of the system is not translation-invariant. to avoid this problem, we redefine robot *i*'s moment generating function to be

$$\varphi(p_i) = \begin{bmatrix} p_{ix}, p_{iy}, (p_{ix} - u_{ix})^2, \\ (p_{ix} - u_{ix})(p_{iy} - u_{iy}), (p_{iy} - u_{iy})^2 \end{bmatrix} \quad (24)$$

where (u_{ix}, u_{iy}) is the robot's estimate of the group center of mass. With this choice of $\varphi(p_i)$, the robots estimate the swarm moments with respect to its center if mass (central second moments) rather than with respect to the origin.

Thirdly, the continuous-time algorithms were adapted to be discrete. The discreet time estimator is written

$$v_i = \overline{v_i} + \gamma(\varphi(p_i) - \overline{v_i}) - K_p \sum_{j \in N(i)} (\overline{v_i} - v_j) + K_I \sum_{j \in N(i)} (\overline{w_i} - w_j) \quad (25)$$

$$w_i = \overline{w_i} - K_I \sum_{j \in N(i)} (\overline{v_i} - v_j) \quad (26)$$

where v_i is the new swarm moments estimate, w_i is the new estimator internal state, $(\overline{v_i}, \overline{w_i})$ is the previous estimator state, p_i is the current robot position, (v_j, w_j) is the most recently received estimator from robot *j*, $N(i)$ is the set of robots that robot *i* has heard from since the last estimator update, and $K_P, K_I, \gamma > 0$ are estimator gains. Because there is no global synchronization of the robots' clocks, each robot runs its control, estimation, and communication processes on a different schedule. After updating its estimate, the robot discards estimator state information received from other robots, since it is considered outdated.

SIMULATION AND RESULTS

For the simulation, Matlab has been used. In this segment the code shall be explained and also some comments will be made.

A. Assumptions

We assume the following properties for the system:

- *TS=0.001 :* is the time unit (time step)
- *S2U=9 :* are the average time steps an agent waits until he updates himself
- *S2DP=50 :* are the time steps ,for which each agent keeps information given from another agent. After *S2DP* time steps it is consider to be updated and be dropped
- *N=7 :* the number of agents being used.
- *Kp,Ki :* the proportional and integral controller parameters
- *desiredForm :* the desirable formation, based on first and seconds moments, that the multi-agent systems want to acquire.
- *gamma :* The forgetting factor of the estimator.
- *gaMatrix :* The symmetric positive-definite global gain matrix $\Gamma \in \mathbb{R}^{l \times l}$ used by the controller.

The modifications mentioned on the methodology section are of great meaning for a successful outcome. The omission of the $-B\dot{p}_i$ term in Eq. (18) is justified once we assume that the overall control of the system is given through the velocity $u = \dot{p}$.

Although this assumption does arise some problems: once we are not able to control acceleration, overshoot is really hard to avoid. Moreover, this can very easily lead to oscillations around the equilibria and in more severe situations in a total unstable system.

In order to surpass this problem we need to assure that the system will make small steps towards the equilibria. This is achieved by setting a speed limit for the agents.

Eq. (24) is also of great value, since is makes the control of the system translation-invariant in the means of seconds order moments. This way, when comparing the estimated second order moments of

each agent with the second order moments of the global well known desired formation the results are the right ones. There is no need to apply Eq. (24) to the global desired moments, since we assume that they are already centered around the first order moments.

Lastly, Eq. (25) and (26) provide us with a more algorithmic way of solving the estimation problem. These Equations are not new since all they do is implementing the gradient descent algorithm with a plus instead of a minus, since we want to get closer to where the gradient is leading us:

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \gamma \nabla F(\boldsymbol{a}_n) \Rightarrow$$
$$\boldsymbol{a}(n+1) = \boldsymbol{a}(n) + \dot{\boldsymbol{a}}(n)$$

B. *Implementation*

The simulation is based in an outer loop. The positions of the agents are updated in every loop and are drawn every *S2D* loops.

Agents are not updated all together. We have inserted a random parameter in order to make the simulation more realistic. In every loop, it is checked whether an agent must be updated or not. If yes, the following procedure is happening.

Agent firstly transmits his packet (that is his estimations w and v) to every agent within a range. The cumulative distribution function used for the successful transmission is the exponential, with a parameter of 1000 :

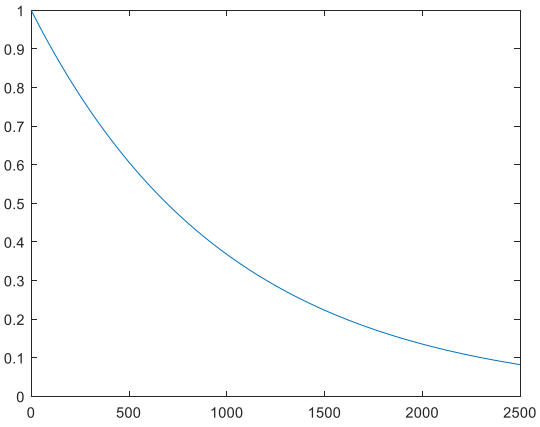

Figure 3

In the above function, again, a random parameter is inserted that overall makes the transmission harder.

Afterwards, the agent checks if he should drop an old packet that he had received previously.

Immediately after, the agent communicates with certain agents in order to calculate his own estimation of the moments of the system.

In the end the agent updates his control state (velocity) by using his local controller. The update is being held with the knowledge of the desirable formation the system must reach.

C. *Results*

Subsequently some diagrams and maps are shown that have been extracted from the simulation.

In the mapping, the grey ellipse illustrates the first and second order moments that the system wants to acquire and the green ellipse the moments of the multi-agent system. As mentioned by [10] there really do exist an optimized ellipse that can illustrate in the best way possible the first and second order moments of a formation. In this paper, we have used an a lot simpler approach, which is robust and efficient and fits better the specifications of the problem. The approach is implemented by the *ellipse_given_Mom()* function.

The bar graph shows the number of connections its agent has at the moment. As a connection is also consider an older packet which hasn't been outdated yet.

The vector diagram shows the velocity of each agent at the moment. As we previously said, the velocity is being clipped if higher than a limit.

The rightmost graphs are expressing the moment errors between the swarm and the goal moments. The Distance Error diagram expresses the first and second moment error together, thus giving us a

distance error. The rest of the diagrams give us the second order moments error.

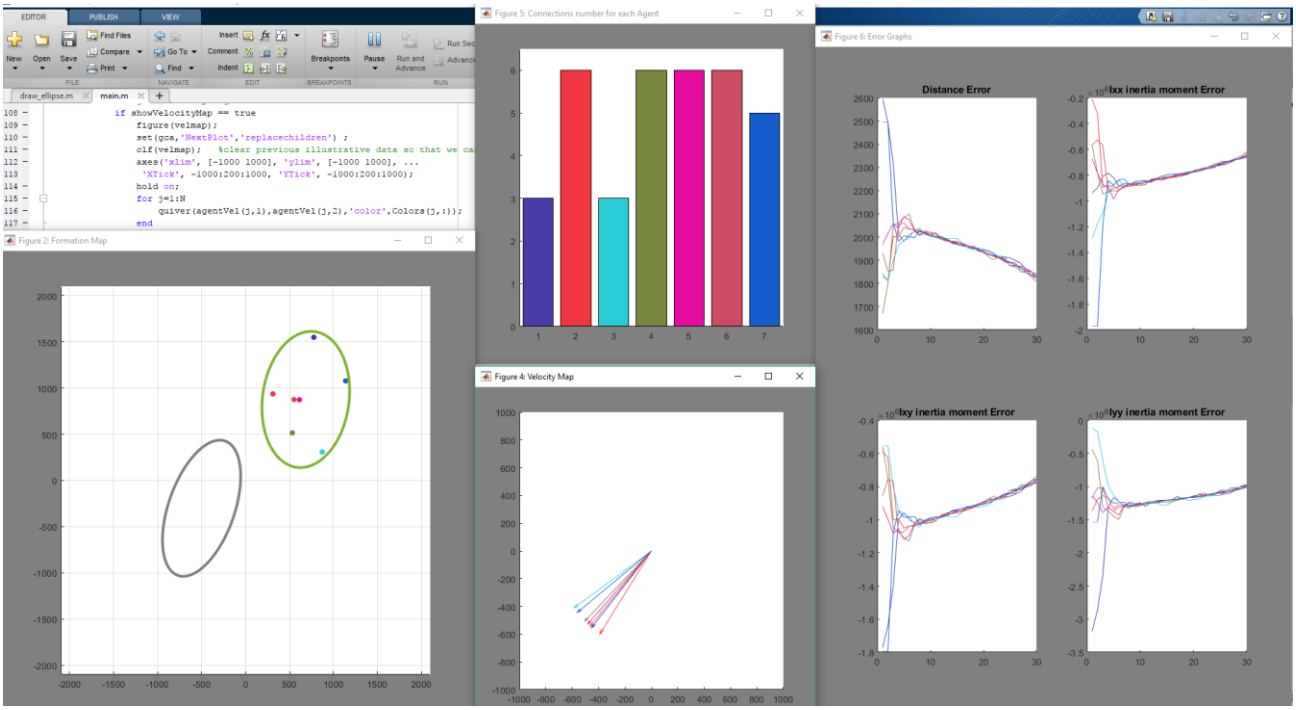

Figure 4 : The beginning of the "actuator".

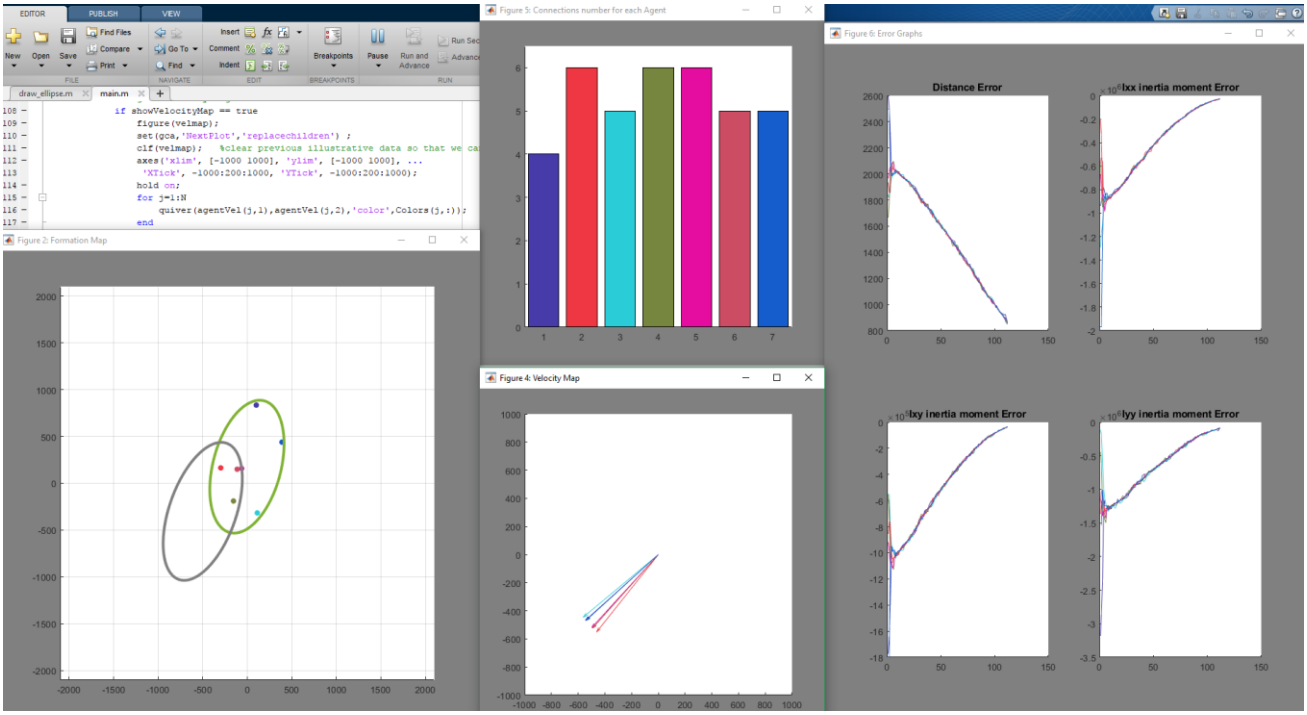

Figure 5 : The swarm is getting closer. Error graphs show the rate of improvement. The swarm connectivity continues on being robust, since agents haven't been distanced a lot yet. All the agents continue to have the same velocity direction.

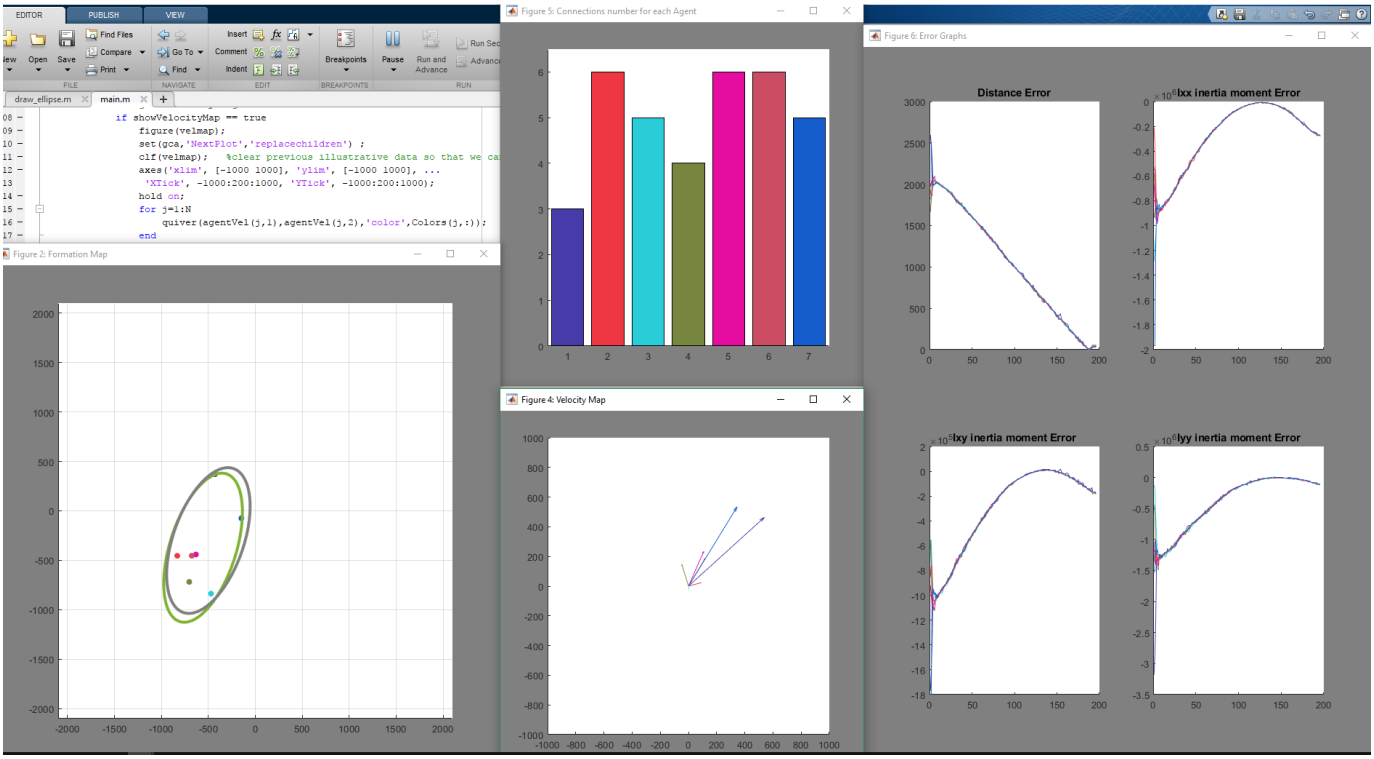

Figure 6 : The swarm has approached the desirable formation in a satisfying way

D. Comments

The steady state error introduced is something that emerges from the imperfect knowledge of the agents towards the global system properties. Given that the agents have perfect knowledge over the system the steady state error is eliminated. This can happen in reality, by assuming that all agents communicate with each other. Then, thanks to the PI Consensus estimator, perfect knowledge can be achieved for every agent and thus have zero steady state error.

The lack of an integrated methodology on calculating the constant and parameters used on the system, has been proven very time consuming. Controller parameters and any other parameters or constants were decided for the system through a trial and error procedure and the authors' intuition from control theory. It would be a pleasure for the authors to someday proceed on the standardization of such procedure.

CONCLUSIONS AND FUTURE SUGGESTIONS

The overall system behavior is considered to be good enough. The system's ability to be independent from a central control authority is the most important one. Agents are autonomous and do not care for not having perfect knowledge over the system, since this knowledge can be estimated through the agents' neighbors. The various estimations of the agents are getting averaged and known to every node in the system. Finally by using a local controller in each agent, the desired formation can be reached.

Below a serious list of enhancements is being mentioned.

A. Well known improvements

A serious consideration is that the graph being created must be and continue to be connected. This constraint is not implemented in the above system but it can be left for future work.

Another problem that arises is the lack of collision avoidance control. The collision avoidance will not only make the system more realistic (since in real life we do not desire robots getting crashed with each other) but will also make the system behave better. That is because without collision avoidance agents are not uniformly distributed in space· in contrary, as time passes by, there is a possibility that some of them form a consistent non-easily-separable mass. This way they hurt the overall system balance and lead the system to poorer performance.

In addition, it is very interesting to see the behavior of the system given that we would like to provide control for the system by means of acceleration.

B. Improvement suggested by the authors

Finally, the authors would like to suggest a new heuristic algorithmic way of improving the system's behavior.

We have noticed that the systems behaves better (as speed of convergence and steady state errors are concerned) when it tries to approximate second order moments given that it has already approximated the first order.

Also it is well known that when connection numbers are higher a better result is being attained.

We suggest a connectivity constrained algorithm where initially the first order moments are being achieved under a robust and strong connectivity. Subsequently the multi-agent system can proceed to approximate the second order moments too.

We believe that by standardizing this procedure we may be able to achieve a faster convergence, smaller steady state error and successful convergence in a higher number of desirable formations and initial states· thus, yelling a most robust system which will have no problem on taking over perpetually[1] missions the one after the other, since every time it reforms itself

[1] We have noticed that for the current system, a limited number perpetual missions can be held, since from a point on onwards the agents are either scattered either welded

REFERENCES

[1] P. Yang, Randy A. Freeman and Kevin M. Lynch, "Multi-Agent Coordination by Decentralized Estimation and Control," p. 17, December 2008.

[2] I. D. Couzin, J. Krause, R. James, G. D. Ruxton and N. R. Franks, “Collective memory and spatial sorting in animal groups,”, Theoretical Biol., vol. 218, pp. 1–11, 2002..

[3] J. K. Parrish, S. V. Viscido and D. Grunbaum, “Self-organized fish schools: An examination of emergent properties,”, Biol. Bull., vol. 202, pp. 296–305, Jun. 2002..

[4] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,”, Comput. Graph., vol. 21, no. 4, pp. 25–34, 1987..

[5] B. Schechter, “Birds of a feather,”, New Scientist, pp. 30–33, Jan. 23, 1999.

[6] V. Kumar and C. Belta, “Abstraction and control for groups of robots,”, IEEE Trans. Robotics, vol. 20, no. 5, pp. 865–875, Oct. 2004..

[7] R. Murray and R. Olfati-Saber, “Consensus problems in networks of agents with switching topology and time-delays, IEEE Trans. Automat. Control, vol. 49, no. 9, pp. 1520–1533, Sep. 2004..

[8] D. P. Spanos, R. Olfati-Saber and R. M. Murray, “Dynamic consensus on mobile networks", in Proc. IFAC Available: http://www.cds.caltech.edu/~demetri/docs/SOM_IFAC05_DC.pdf, 2005.

[9] R. A. Freeman, P. Yang and K. M. Lynch, “Stability and convergence properties of dynamic consensus estimators” pp. 338–343., in Proc. IEEE Int. Conf. Decision Control, Dec. 2006.

[10] B.B. Chaudhuri and G.P. Samanta, "Elliptic fit of objects in two and three dimensions by moment of inertial optimization," p. 7, January 1991.