Weighted Average Consensus Algorithm for Estimating Histogram within Distributed Robots

Yang Li[1] and Joe Jackson[2]

Abstract— **In a centralized scheme, each robot sends its data to a central center. This is extremely challenging if the network topology changes. What if there is no single component with global knowledge, no system element with responsibility for controlling the computation? In a distributed scheme, there is no central center and the robots do not have any global knowledge of the network topology. Each robot only exchanges data with its neighbors and carries out local computation. The goal is for each sensor to eventually have a good estimate of the unknown whole picture. In this paper, we focus on the design and analysis of a weighted average consensus algorithm. To test the proposed algorithm, given an image, we separate it by N parts, each representing a distributed robot. The neighbors of one robot depend on their spatial positions in the image. And the histogram information of the robot can be sent to the neighbors to estimate the histogram of the whole image which is the final goal of the algorithm.[1], [2], [3]**

I. INTRODUCTION

Traditionally, the study of distributed computing typically assumes the existence of a logically centralized spatial information repository, capable of collating all the relevant spatial information and controlling the computational procedure[4]. But what happens when we relax this assumption? How can we compute in spatial information systems where there is no single component with global knowledge, no system element with responsibility for controlling the computation?

This rather conservative trivial approach - the centralized approach - was quickly dismissed because it was highly inefficient, poorly salable, and dangerously insecure. The advent of mobile and wireless communication has shown to researchers that the universe of distributed computing is indeed much wider than previously thought. In particular, spatial information is central in many new distributed computational environments such as mobile sensor networks, delay-tolerant networks, mobile ad hoc networks, and swarms of autonomous mobile robots.

Decentralized distributed computing is the study of decentralized techniques for computing with spatial information. Distributed computing usually requires the combination of information from distal geographic location, a challenge in the context of spatial constraints on the movement of information.

Consensus problems have a long history in computer science and form the foundation of the field of distributed computing [5]. Recently, much attention is paid to the problems related to multiagent networked systems which have strong connection with consensus problems. The topics included are consensus, sensor fusion, random networks, collective behavior of flocks and swarms, formation control for multirobot systems and so on.

Papers that have been instrumental in paving the way for more recent advances in study of self-organizing networked systems, or swarms. These networked systems are comprised of locally interacting mobile/static agents equipped with dedicated sensing, computing, and communication devices.

Distributed computation in sensor network is a well-studied field. Average consensus computes iteratively the global average of distributed data using local communications [6]. There are many versions and extensions of basic consensus [7], [8].

Distributed Sensor Fusion in Sensor Networks: the most recent application of consensus problems is distributed sensor fusion in sensor networks. This is done by posing various distributed averaging problems require to implement a Kalman filter, approximate Kalman filter, or linear least-squares estimator as average-consensus problems. Novel low-pass and highpass consensus filters are also developed that dynamically calculate the average of their inputs in sensor networks. In distributed computing, the consensus problem seems to be one of the central topics which has attracted intensive research. So why is the consensus problem so important? What can we achieve with consensus both in theory and in practice? In paper [9], it shows that there is no asynchronous deterministic consensus algorithm that tolerates even a single crash fault. Note that in the synchronous setting, there is a deterministic algorithm that terminates in $f + 1$ rounds when $\leq f$ processes crash. One important application of the consensus problem is maintaining consistency in a distributed network: Suppose that you have different sensor nodes monitoring the same environment. In the case where some of these sensor nodes crash (or even start sending corrupted data due to a hardware fault), a consensus protocol ensures robustness against such faults.

In this paper, we do research on a well known consensus algorithm called weighted average consensus algorithm. Proposed a simple distributed, iterative scheme to compute an estimate at each robot based on average consensus over the whole network. The rest of the paper is stated as follows. In section II, we will describe the algorithm to be tested. Then section III shows the details of the experiment and then the results we get from the experiment. In section IV, we analyze the algorithm. Finally, the conclusion is given and some possible future work is detailed.

[1,2]Yang Li and Joe Jackson are with the Department of Computer Science, University of Colorado Boulder, CO, USA yang.li-4 at colorado.edu and joseph.jackson at colorado.edu

II. WEIGHTED AVERAGE CONSENSUS ALGORITHM

In this section, we describe the average consensus algorithm step by step. Based on the notation introduced, we try to give a algorithmic frame. The proof of convergence will be skipped since our main focus is introduce the algorithm and apply it on estimating histogram of an image.

A. Algorithm

In the scheme, each robot maintains a fixed and small storage memory, and exchanges information with its neighbors in a very simple and isotropic manner [3]. It diffuses information across the network by updating each robots data with a weighted average of its neighbors. At each step, every robot can compute a local weighted estimate, which eventually converges to the global estimate.

We model the topology of a sensor network by an undirected graph the communication graph. Let $G = (E, V)$ denote an undirected graph with vertex set $V = \{1, 2, ..., n\}$ and edge set $E \in \{(i, j)|i, j \in V\}$, where each edge (i, j) is an unordered pair of distinct nodes. A graph is connected if for any two vertices i and j there exists a sequence of edges (a path) $(i, k_1), (k_1, k_2), ..., (k_{s1}, k_s), (k_s, j) \in E$. We represent the time-varying communication graph of a sensor network by $G(t) = (E(t), V)$, where $E(t)$ is the set of active edges at time t. Let $N_i(t) = \{j \in V|(i, j) \in E(t)\}$ denote the set of neighbors of node i at time t, and $d_i(t) = |Ni(t)|$ denote the degree (number of neighbors) of node i at time t.

We use a distributed linear iterative method to compute the average. At $t = 0$ (after all sensors have taken the measurement), each node initializes its state as $x_i(0) = y_i$. At each following step, each node updates its state with a linear combination of its own state and the states at its instantaneous neighbors (nodes that belong to $N_i(t)$).

Although we are interested here in time-varying graphs, it is interesting to first consider the case when the communication graph is fixed, and we use a time-invariant weight matrix W.

In the time-varying case, we determine the weight matrix at each step as a function of the instantaneous communication graph. In particular, we focus on the following two simple rules for choosing the weights: Maximum-degree weights and Metropolis weights.

Maximum-degree weights. Here we use the constant weight $1/n$ on all the edges, and choose the self-weights so that the sum of weights at each node is 1.

Metropolis weights. The Metropolis weight matrix is defined as:

$$W_{i,j} = \begin{cases} \frac{1}{1+max\{d_i,d_j\}} & \text{if } (i,j) \in E, \\ 1 - \sum_{(i,k)\in E} W_{i,k} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

With Metropolis weights, the weight on each edge is one over one plus the larger degree at its two incident vertices, and the self-weights are chosen so the sum of weights at each node is 1.

The Metropolis weights are well suited for distributed implementation, since each node only needs to know the degrees of its neighbors to determine the weights. The nodes do not need any global knowledge of the communication graph, or even the number of nodes n. For the maximum degree weights, each node has to know n, or at least an upper bound of n.

Having all the above information, we are ready to introduce the algorithm framework in Algorithm 1.

Algorithm 1 Average Consensus Algorithm

Input: Graph $G = \{V, E\}$, $n = |V|$ robots R
Output: n robots with updated data
Initialization:
1: Initialize R with data stored in the memory
2: Compute the degree of each robot d_i
3: Initialize the weights matrix W according to graph G
4: Compute the average data of all the robots, $R_average$
Loop Process:
5: **for** $t = 1$ to T **do**
6: Update weights matrix W if necessary (if G is updated)
7: **for** $i = 1$ to n **do**
8: $R_i = W_{i,i}R_i + \sum_{j\in N_i(t)} W_{i,j}R_j$
9: **end for**
10: **end for**
11: **return** n robots R with updated data stored in the memory

In line 1: n robots are initialized with the data they obtain. In our experiment in Section III, the data in each robot is the histogram information of the small part of image, i.e., three vectors of red, blue, green frequency values.

In line 2: the degree of each robot is the number of neighbors it has in the graph.

In line 3: the weights matrix W is initialized based on the graph input. Equation 1 is used to compute each $W_{i,j}$.

In line 4: the average data is used to decide if the algorithm converges. In our experiment, it is the RGB vectors of the whole image we input.

In line 5: the termination criteria is the maximum number of iterations. An alternative one is the threshold as the difference between RGB vectors of one robot and the whole image

In line 6: it is necessary for time-varying graph/network ,since in the experiment, we use time-invariant weight matrix W, that is the communication graph is fixed.

In line 8: At each inside iteration, each robot updates its data with a linear combination of its own data and the data at its instantaneous neighbors (robots that belong to $N_i(t)$). In the experiment each N_i doesn't change as t increases.

In line 11: The data for the n robots is returned. In the experiment, we expect that the RGC vectors of each robot are similar with those of the whole image.

B. Convergence Proof

We skip it for the report. (Matrix theory, algebraic graph theory related)

In paper [3], the authors show that both the maximum-degree and Metropolis weights guarantee convergence of the average consensus provided that the time-varying communication graphs are connected in a long run; more precisely, if the infinitely occurring communication graphs are jointly connected.

III. EXPERIMENTS AND RESULTS

A. Experiment Setup

In the experiment, histogram of an image is used as the data to be estimated. For each image we can extract the RGB information, by adding three arrays of RGB value together, we obtain a general array of values as the data. We can separate an image into N parts, each part representing a robot. The histogram information of each part of the image is the data.

Here we use the most commonly used image called "Lena" (see Figure 1) to extract the data for each robot, and we separate the image into $8 \times 8 = 64$ parts (see Figure 2). The communication graph is defined by the spatial position and each image as a node in the graph is only connected to those in the eight directions. In this sense, there are three kinds of nodes with different number of neighbors, that is Corner nodes, Frame nodes and Inner nodes (illustration is shown in Figures 3, 4, 5).

Fig. 1. Lena

The RGB histogram information are show in Figure 6. Since the histogram information is only regarded a kind of data that is needed to be estimated, the RGB information are mixed together to form the overall histogram. The number of pixels with the same values in RGB arrays are added together and finally all the four histogram arrays are normalized as probability arrays (all the values in an array sum up to 1).

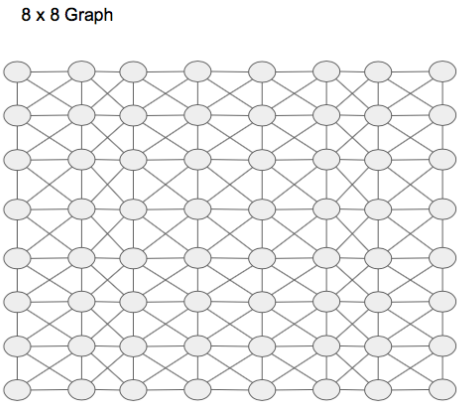

Fig. 2. Connection Graph

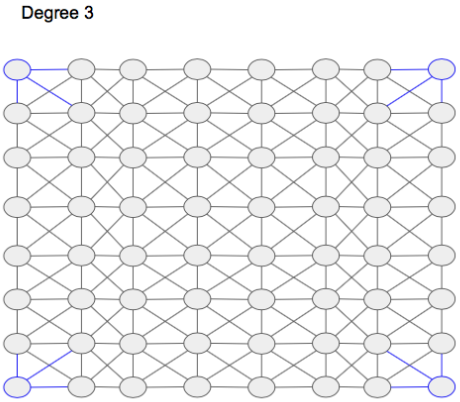

Fig. 3. Connection Graph with Corner Nodes (Blue)

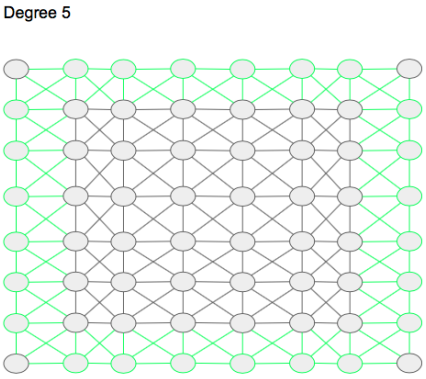

Fig. 4. Connection Graph with Frame Nodes (Green)

B. Results

We apply this proposed algorithm on the image, having the overall histogram information of Lena as the data to be estimated. The metric of convergence we used is the Eulerian Distance from histogram array of the whole image to histogram arrays of all nodes. Based on the metric, the termination criteria of the algorithm is when all the Eulerian Distances are less than a threshold (here we use 0.02).

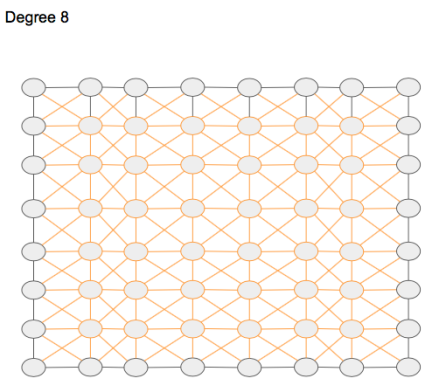

Fig. 5. Connection Graph with Inner Nodes (Orange)

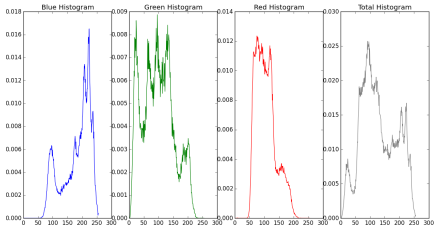

Fig. 6. RGB and overall histogram information of Lena

In Figure 7, the Eulerian Distance of all the 64 nodes during the running time are shown. We can see that the algorithm converges after about 20 iterations.

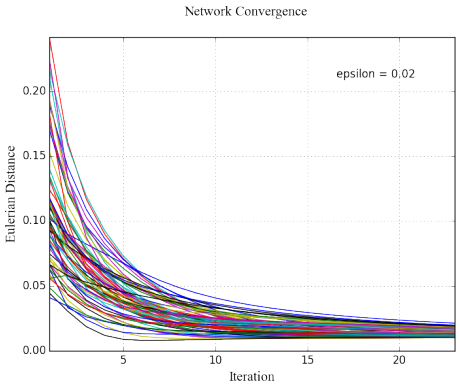

Fig. 7. Convergence of all nodes

From the setup section, we know that there are different types of nodes. Figure 8 shows all the convergence plots of different types. It is easy to find out that Inner nodes quickly converge after about 7 iterations, while it takes Corner and Frame nodes more than 12 iterations to converge.

Specifically, we do experiments on each of the types of nodes, choosing certain ones. For example, we choose node 1 for Corner node, node 9 for Frame node and node 26 for Inner node. The convergence of their histogram are shown in Figures 9, 10 and 11. In each figure, the left histogram (in blue) is the original histogram of each node, the middle histogram (in red) is the one just before the algorithm did the last iteration and the right histogram (in gray) is the last one whose Eulerian Distance to the whole histogram is less than 0.02.

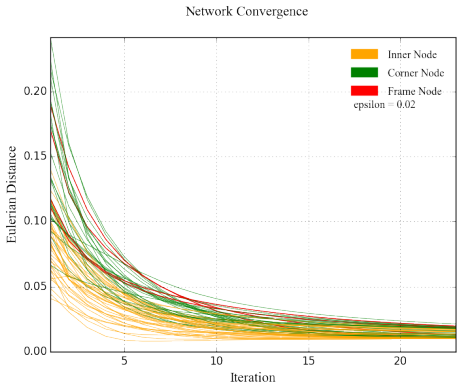

Fig. 8. Convergence of all nodes of different kinds

In Figure 9, it takes the Corner node (node 1) 13 iterations to converge. In Figure 10, it takes the Frame node (node 9) 12 iterations to converge. And in Figure 11, it takes the Inner node (node 26) 7 iterations to converge.

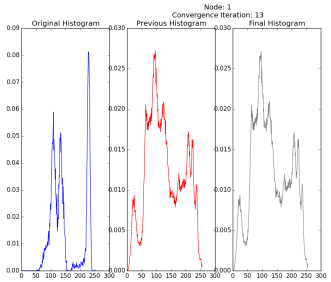

Fig. 9. Convergence of Corner Node with histogram

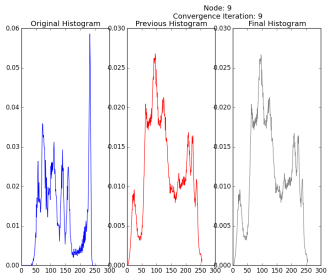

Fig. 10. Convergence of Frame Node with histogram

IV. DISCUSSION

In the experiment, we empirically proved that the average consensus algorithm can guarantee convergence. It is like a physical phenomenon that different part of a metal has different temperature and the thickness of the metal is not uniform. As time goes, the temperature of all parts of the metal will converge to be the same.

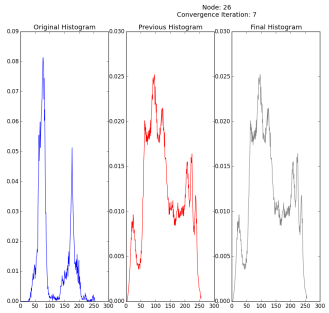

Fig. 11. Convergence of Inner Node with histogram

We can find out that the degree of a node in the graph can influence the speed of convergence. Generally speaking, the larger the degree a node has, the faster the node converges.

V. CONCLUSION

Distributed computing plays an important role in a distributed environment where there is no central element could be responsible for the global computing and have global knowledge of the whole communication graph. Consensus problems is definitely a core part of distributed computing. In this paper, we investigated a simple yet convergence-guaranteed consensus algorithm which is commonly used in sensor network fusion. We first introduced the general algorithm framework of this distributed average consensus algorithm after the concepts of graph and degree of a node. Then, we used different parts of an image to represent 64 nodes of a graph, defining the degree of a node by the number of its spatial neighbors. The histogram information of each part of the image is used as the data to pass through the graph. The goal of the average consensus algorithm for this task is to have the histogram of all nodes reach a consensus, i.e. histogram of all parts in the image converges to that of the whole image. Experimental results shows that the distributed average consensus algorithm guarantees the convergence. We also discussed different types of nodes have different performance of convergence speed. The results we came with is that a node of larger degree converges faster than one of smaller degree.

REFERENCES

[1] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 5492–5498, IEEE, 2007.

[2] R. Olfati-Saber, A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 63–70, IEEE, 2005.

[4] M. Duckham, *Decentralized spatial computing: foundations of geosensor networks*. Springer Science & Business Media, 2012.

[5] N. A. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[6] S. Kar and J. M. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 355–369, 2009.

[7] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH computer graphics*, vol. 21, pp. 25–34, ACM, 1987.

[8] R. O. S. R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the 2003 American Controls Conference*, 2003.

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.