Pipelined Consensus for Global State Estimation in Multi-Agent Systems

Paper 19

ABSTRACT

This paper presents *pipelined consensus*, a practical, robust consensus algorithm for multi-agent systems using mesh networks. During each round, each agent starts a new consensus. Each agent maintains the intermediate results for the previous k consensus in a *pipeline* message. After k rounds, the results of the first consensus are ready. The pipeline keeps each consensus independent, so any errors only persist for k rounds. This makes pipelined consensus robust to many real-world problems that other algorithms cannot handle, including message loss, changes in network topology, sensor variance, and changes in agent population. The algorithm is fully distributed and self-stabilizing, and uses a communication message of fixed size. We demonstrate the efficiency of pipelined consensus in two scenarios: computing mean sensor values in a distributed sensor network, and computing a centroid estimate in a multi-robot system. We provide extensive simulation results, and real-world experiments with up to 24 agents. The algorithm produces accurate results, and it handles all of the disturbances mentioned above.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Design, Experimentation

Keywords

Linear Average Consensus, Pipelined Consensus, Communication Error, Convergence Time, Centroid Estimation

1. INTRODUCTION

Constructing local estimates of global state is a critical utility in multi-agent systems. For example, in an environment monitoring scenario where a group of mobile agents are deployed to collectively cover a large interested area [15], we may be curious about the mean measurement value gathered across all the agents to decide if something unusual or dangerous is happening. In a social network of human agents [6], we are interested in how global ideas and opinions form, spread, and cluster. Knowing the global state information can help agents coordinate more efficiently. Flocks of

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

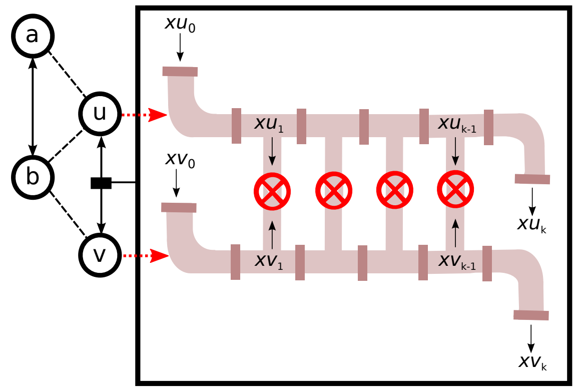


Figure 1: A network of agents performing pipelined consensus, an extension to pair-wise gossip-based consensus. Each node stores k values in their pipeline, which is a queue containing multiple concurrent consensuses. In this example, agents u and v are in the middle of gossiping. At the beginning of a gossip, a new value is inserted into the pipeline, and older values are shifted along the pipeline. The two agents then perform consensus on each cell of their pipelines. The oldest value in the pipeline is taken as the current estimate of the global value. Pipelined consensus is a robust practical solution to global state estimation problems in multi-agent systems.

birds, schools of fish, and crowds of people measure the motion of the nearby neighbors in order to achieve large group-level formations [19]. In a network of mobile agents, sharing local measurement probability information can be used to produce a more informative configuration of the positions of the nodes [8]. In the work on cooperative transportation by a group of robotic agents [20], inferring the forces from other agents let each individual agent know how to align its own force with others' to achieve efficient cooperation.

These examples all require global state estimates in distributed multi-agent systems. The simple approach of broadcasting the local state of each agent will eventually use all available communication bandwidth as population increases. Consensus algorithms can compute global estimates using fixed-size messages, but with few exceptions [16], are not robust to real-world errors, such network topology changes and communication errors, except where specially designed.

In this paper, we present the *pipelined consensus* algorithm. It can continuously track a changing global mean while being robust to disturbances such as initial conditions, topology changes, sensor errors, and communication failures. The core idea behind our algorithm is that the agents start a new consensus each round; initializing them with new measurements of local state. Older consensus estimations are stored in a pipeline queue in order. Consensus estimations corresponding to similar start times will evolve together in every communication round, largely independent of estimations belonging to different start times, as shown in Figure 1. A pipeline size, k, must be chosen such that the estimation will approach the true average before the pipeline runs out of space. This is the fundamental trade-off of our approach: the pipeline size, k, must be larger than the natural convergence time of consensus in the network, τ. Larger values for k allow us to handle larger, more complex networks, but use more communication bandwidth.

In this paper, we study the convergence time τ for a large set of mesh networks, and demonstrate that pipelined consensus is a feasible solution for the vast majority of them. We then present two examples and many experiments to demonstrate the robustness properties of the algorithm : computing the global mean of a quantity in a distributed sensor network, and estimating the centroid of a group of robots.

In multi-agent manipulation tasks, mobile agents require knowledge of the geometry of the object they are transporting in order to properly manipulate the object. The centroid is a value of the object that is estimatable by agents using only their own positions. If the agents surround the object, the centroid of the object can be approximated by the average of the positions of all agents. Simulations and hardware experiments are performed in this paper to show that the centroid can be found even without a global reference frame.

There is a large literature regarding consensus in multi-agent systems. The original fundamental work, [7, 11] showed that all agents in a group could agree on a common value only by communicating their neighbors in the network. Our work uses this basic approach, but runs multiple consensus in parallel. Many applications of consensus use a value that could be constantly changing, such as an environmental sensor or other controlled signals. To handle this, the concept of dynamic consensus was introduced in [17, 18] and convergence was proved. More generally, [3, 21] used a proportional-integral filter on the typical consensus algorithm to achieve robust average tracking of time-varying inputs regardless of the initial states. In both dynamic and robust consensus, the input signals to the nodes must be known, whereas our pipelined consensus algorithm does not have this requirement. One important aspect of tracking a changing consensus is the rate of convergence. [1, 4, 12, 13] derived upper bound on the convergence time under different assumptions and from different prospective. We apply our pipelined consensus algorithm to estimate the changing centroid of an object as the robots move. Similar work on the stationary centroid estimation using only local reference frames can be found in [2, 5]. Handling communications errors is difficult for most consensus algorithms. The work in [16], injects a portion of the input signal into the state estimate of each robot during each round. This lets the estimated global mean on each agent be robust to communication errors, but at the cost of a large variance across individual estimates. Our approach produces qualitatively similar results, but without the additional variance causes by re-injecting the sensor value into the local estimate.

The rest of the paper is organized as follows. We describe our model and assumptions in Section 2. The pipelined consensus is formally proposed in Section 3. Extensive simulation and experiment results are provided in Section 4, where we use pipelined consensus to track a changing average and perform centroid estimation. Convergence time and tracking error subject to different disturbances are also analyzed. We discuss and conclude our work in Section 5 and 6.

Algorithm 1 Pipelined Consensus

```
 1: P_i ← NULL                         ▷ Initialize all pipeline cells with null values
 2: Repeat forever on each robot u
 3: Randomly choose a neighbor v ∈ N(u) to gossip
 4: if |P_u| = k then                  ▷ |P_u| is the number of non-null cells
 5:     REMOVE(P_u)                    ▷ Remove the oldest value
 6: end if
 7: INSERT(P_u, xu_0)                  ▷ Insert new input value
 8: for t = 0 : MIN(|P_u|, |P_v|) do   ▷ Perform consensus on pipeline
 9:     xu_t = CONSENSUS(xu_t, xv_t)
10: end for
```

2. MODEL AND ASSUMPTIONS

We assume that our agents are in an environment too large for centralized communication. A communication network is built by the agents using inter-agent communications between nearby agents within a fixed distance d, where d is much smaller than the size of the environment. Each agent constitutes a vertex $u \in V$, where V is the set of all agents and E is the set of all agent-to-agent communication links. n is $|V|$, the total number of agents in the network. We model the agent's communications network, $G = (V, E)$, as an undirected unit disk graph. We assume that G is connected. The neighbors of each vertex u are the set of agents within line-of-sight communication range d of agent u, denoted $\mathcal{N}(u) = \{v \in V \mid \{u, v\} \in E\}$.

We model algorithm execution as proceeding in a series of discrete *rounds*. While actual operation in many practical systems is asynchronous, implementing a synchronizer simplifies analysis greatly and is easy to implement [9]. In this paper, we focus on the convergence time of consensus τ measured by the number of rounds of computation required to achieve some ε-bound on error.

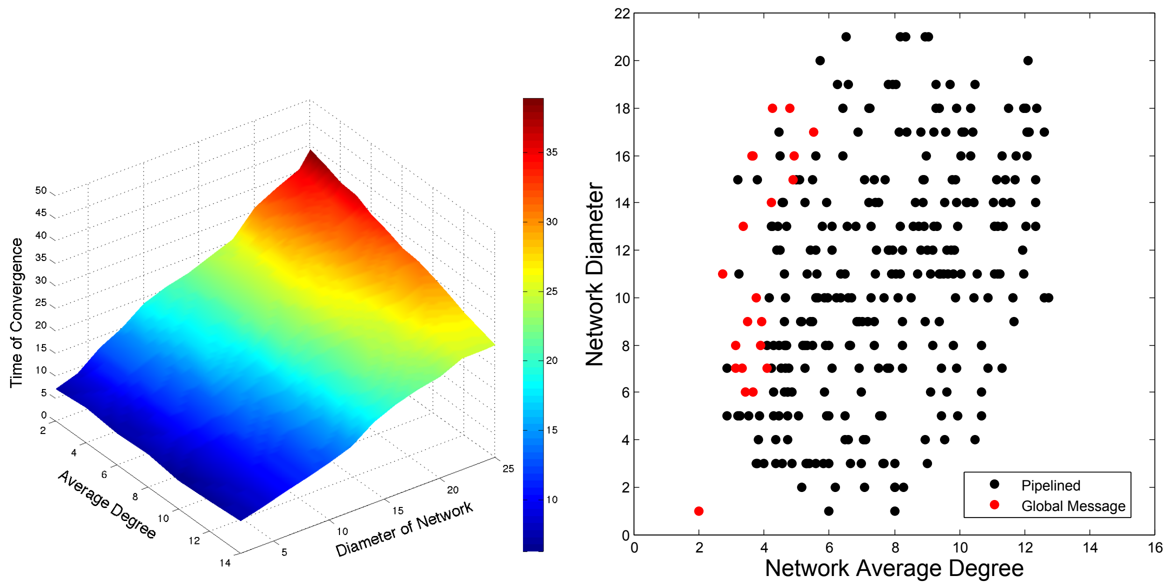
We assume our agents are homogeneous and are modeled as a disk. Each agent is situated at the origin of its own local coordinate frame with the $\hat{x}$-axis aligned with its current heading. Agents can measure the relative pose of its neighbors. The relative pose between two agents u and v is given by three measurements, bearing B_{uv}, orientation O_{uv}, and range R_{uv}. Bearing is the angle from agent u's heading to agent v's relative position. Orientation is the angle of agent v's heading from B_{uv}. Range is the distance between agent u and v.

3. PIPELINED CONSENSUS ALGORITHM

3.1 The Algorithm

Pipelined consensus is based upon linear average consensus, where in which agents continuously update their value to the average of their estimates. In pipelined consensus, each agent stores k values instead of only one value (See Figure 1). Pipelined consensus is described in Algorithm 1. At first, every agent's pipeline, P_i, is initialized with null values(Line1). Pipeline values are updated each round of successful pair-wise gossip with a neighbor. During a gossip, each agent first checks the size of non-null cells in the pipeline, $|P_i|$. If the pipeline is filled with values, the oldest value is removed from the tail of the pipeline(Lines 4-7). This keeps the size of the pipeline constant at k, as values are always inserted. The current input value, xi_0, is inserted to the head of the pipeline. Next, consensus is performed on each respective cell in the pipelines.

Note in Line 8 of Algorithm 1, we take the minimum of the size of the two pipelines. Consensus is only performed on values that



(a) (b)

Figure 2: (a) Viability of algorithms under networks of different degrees and diameters. 300 networks were sampled for 20 trials each. The red dots show a region where a global message is preferred, while the black dots show where pipelined consensus is preferred. Algorithm preference was determined by message size. If the pipeline's size k was greater or equal to n, the size of the network, a global message was preferred. Otherwise, pipelined consensus was preferred because it needs the smaller message size. The majority of results showed that pipelined consensus was preferred. **(b)** Plot of convergence time for 10% error as a variable of diameter and degree of a network. Convergence time τ is in number of rounds. The same networks as in **(a)** were used, 20 trials each. The time of convergence is strongly related to the connectivity of the network. Networks with a large average degree have a smaller τ even in the network with high dimension. However, the diameter in the network with the lower degree affects on the time of convergence, such that the network with low degree and high dimension have the larger τ.

can be paired between the two pipelines, bounded by the minimum size of the pipelines.This is done because equal pipeline sizes are not guaranteed. A feature of pair-wise gossip algorithms is that for each round, every agent may or may not preform a consensus. As values are only inserted upon successful pairings and consensuses, the pipeline sizes are not always equal. This is done to be more efficient with our sampling and message size. Inserting a new value every round where consensus may not occur would dramatically increase the size of k. By accepting different sizes of pipelines, our message size stays reasonable. Disadvantageous network configurations and poor random choices can lead an agent to not perform consensus for a long time. However, for applications that sample values and errors from time independent distributions, the effects of this desynchronization are irrelevant.

3.2 Convergence Time

Selecting an appropriate pipeline size k is key in piplelined consensus. If k is not large enough, the final estimate from the pipeline for any round will not have sufficiently converged. This gives a larger error and variance in the estimate. Based on the convergence time of the network τ, we select a pipeline size k to to reach a desired amount of convergence. However, large values of k have a large message size which is a limiting factor on real communication systems with limited bandwidth. In order to find a balance between τ and k, we analyze the convergence time of different networks to understand τ and k more thoroughly. Figure 2(a) shows the time of convergence for the same sample of networks used in Figure 2. This result illustrates the relation between the degree and diameter of the network to the convergence time of the network.

In pipelined consensus, each agent shares the contents of its pipeline with its neighbors, which is a message of size k, the pipeline size. If k is equal to or greater than n, the number of agents in the network, then there is no advantage in using the pipelined consensus algorithm. In cases such as this, agents could use a *Global Message* to share its estimate with other agents. This message would consist of the value from each agent network, from which the global value could be obtained. We tested the pipelined consensus algorithm on a large sample of networks of varying degree and diameter to show the viability of the algorithm as a solution to global state estimation. As Figure 2(b) shows, pipelined consensus is more practical than a global message in the majority of cases. However, the global message is more useful in networks with low degree and high diameter. This is because consensus performs poorly on networks with weak connectivity and a large path between two agents.

4. EXPERIMENTS

We tested pipelined consensus on both simulated and physical platforms. For our physical experiments, we used the r-one robot as our platform. These robots use an infra-red communication system to communicate with their neighbors in synchronous rounds of 1500 milliseconds. The communication system also measures the relative pose to each neighboring robot (see Section 2). We tested pipelined consensus in two applications: Linear Average Consensus and Centroid Estimation. Pipelined consensus is also highly robust to change in values, sensor errors, population changes, and communication failures. All of these are a feature of real appli-

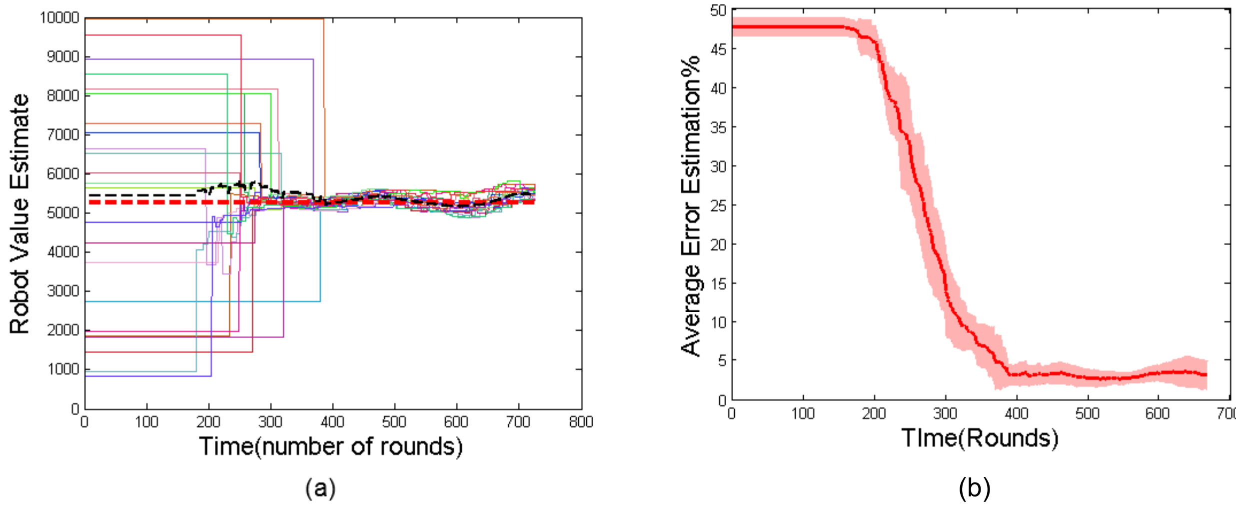


Figure 3: **(a)** Pipeline consensus for a network with 24 agents, with an average degree of 5 and diameter of 5. The average error of their estimate is 2.07%. Agents only give their estimate after the pipeline has been fully filled, which gives the leading edges of their input values at the start of the experiment. **(b)** The result of pipelined consensus for 6 trials of the same experiment in **(a)**. We take the average of each robot's estimate in each trial. The solid red line shows the average of the average errors over 6 trials. The standard deviation of the average error is shown by the shaded red area. This result shows that the error decreases to 2% with pipelined consensus with small variance.

cations and can be handled by our algorithm. We show how our algorithm reacts to these effects in both simulations and physical experiments.

We tested pipelined consensus on both simulated and physical platforms. For our physical experiments, we used the r-one robot as our platform. These robots use an infra-red communication system to communicate with their neighbors in synchronous rounds of 1500 milliseconds. The communication system also measures the relative pose to each neighboring robot (see Section 2). We tested pipelined consensus in two applications: Linear Average Consensus and Centroid Estimation. Pipelined consensus is also highly robust to change in values, sensor errors, population changes, and communication failures. All of these are a feature of real applications and can be handled by our algorithm. We show how our algorithm reacts to these effects in both simulations and physical experiments.

4.1 Pipelined Linear Average Consensus

4.1.1 Physical Experiment, Ideal Conditions

We validated the pipelined consensus algorithm on real robotic agents. We distributed 24 agents on the floor in a grid shape of 4 by 6, which creates a network with the average degree of 5 and a diameter of 5. We used a basic linear average consensus for the gossip protocol in the pipeline. Each agent starts estimation from some constant random initial value. This experiment was carried out under the most ideal conditions possible on the physical platform. However, sensor errors and communication failures inherent to physical systems still occurred. Figure 3(a) shows the estimation progress in each agent in an example experiment. The pipeline size k for this experiment is 20. The result shows how pipelined consensus produces estimates on all robots that have a mean error of approximately zero, but with some variance.

We ran 6 trials of this experiment. Figure 3(b) summarizes the result in tracing the mean of the error estimate with deviations over these trials.

4.1.2 Simulated Experiment, Communication Error

In a basic consensus, a single measurement is taken and used by the agents to come to agreement. This means that any communication errors that occur can lead to a large divergences in the estimated value. Pipelined consensus solves this problem by constantly inputting and pushing values out of the pipeline, so that values that contain error are flushed out after a maximum k rounds. Figure 4(a) shows how communication errors effect both pipelined consensus and basic consensus. Here it is shown that basic consensus converges to a value with very small variance but with an offset from the actual global mean. On the other hand, pipelined consensus continually estimates values with some variance around the global mean with an average error of approximately zero. The variance demonstrated by pipelined consensus can be decreased by increasing the size of k. Pipelined consensus can tolerate communication errors, but an increase in communication errors requires a larger time to converge to an adequate value. Pipeline size increases as communication error increases. Figure 4(b) shows the effects of increasing communication error to the size of a pipeline for different percentage error requirements.

4.1.3 Physical Experiment, Changing Value

As pipelined consensus is robust to changing values, it can track a changing input signal. This feature can be used to calculate the average of a sensor value across a collection of nodes spread out in an environment, such as temperature or light. We tested this feature by having our robots measure the ambient light over time. We changed the intensity of the light by turning on and off the overhead lights in the lab in 20 minute intervals. Figure 5 shows the result of light signal tracking by 20 robots. This figure illustrates the ability to successfully track a changing input signal with accuracy. It also shows the algorithms tolerance to sensor errors. In this experiment, one robot had a very large bias in the light sensor value. However, the converging values still reached consensus over the network.

4.2 Pipelined Centroid Estimation

In tasks of multi-robot manipulation, knowledge of the geometry

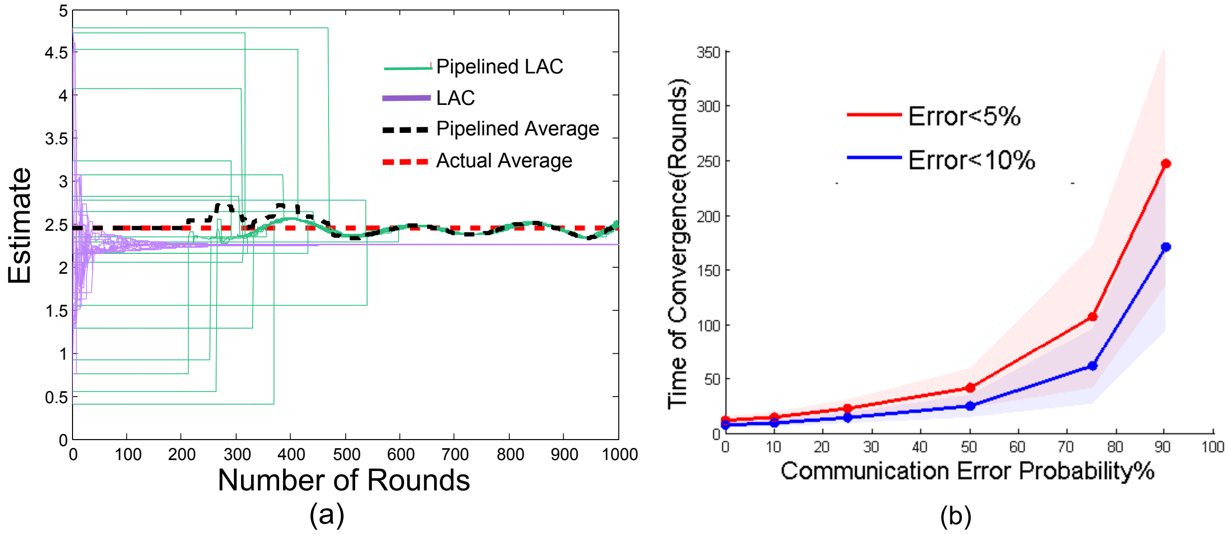


Figure 4: **(a)** Comparison of pipelined consensus and basic consensus in a network of 20 robots with 75% communication error. The size of k in this experiment was 100. The communication errors cause the variance of the result from the pipelined consensus to increase, but the estimated mean remains stable around the actual mean. Basic consensus converges to an erroneous value and remains. **(b)** The effect of communication error in the range of 0% to 90% on the time of convergence with error in the consensus estimate less than 10% (blue solid line) and less than 5% (red solid line). The network is of 72 agents. This result shows the average and standard deviation of the convergence time over 12 trails.

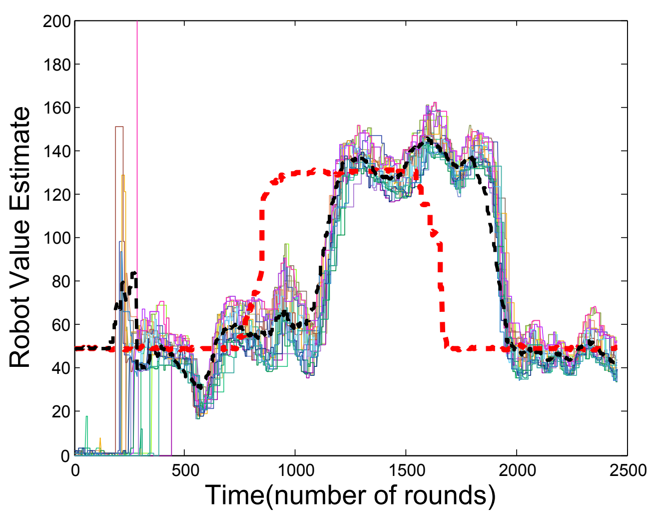


Figure 5: Light tracking experiment by 20 robots. The read sensor value was averaged together. The light began off for 20 minutes, was then turned on for 20 minutes, and then back off for the final 20 minutes. Note the delay of a factor of k rounds in the read input signal to the estimated value. The red dashed line shows the average of light measurement on 20 robots. The black dashed line shows the average of reported values from the consensus on robots. Solid lines show each robot's estimation. One robot had a large bias in reading the light intensity. As a feature of consensus, this value washed out after the consensus occurred.

of the object can be critical. The centroid is a computable value of the object purely from robots' positions along the object, and therefore a good fit for use of our algorithm. In this case, the centroid can be approximated by taking the average of the positions of all the robots. We assume that the robots are distributed around the object in such a configuration that they approximate the shape of the object.

The challenging part is that we assume there is no global reference frame, which means that the robots need to do the estimation in different local reference frames. Therefore, the resulting consensus values are all different for each robot, although they correspond to a common point in the global frame. However, it is impossible for the robots to actually know this in our distributed system. In order to communicate values from one robot to another, a coordinate transformation between different local reference frames must be used. Consider a point x in the global reference frame. Its coordinates in robot u's and v's local reference frame are denoted by ${}^{u}x$ and ${}^{v}x$ respectively. The relationship between ${}^{u}x$ and ${}^{v}x$ is given by

$$ {}^{u}x = {}^{u}_{v}M\,{}^{v}x, \tag{1} $$

where

$$ {}^{u}_{v}M = \begin{bmatrix} \cos(\theta_{vu}) & -\sin(\theta_{vu}) & d_{vu}\cos(B_{vu}) \\ \sin(\theta_{vu}) & \cos(\theta_{vu}) & d_{vu}\sin(B_{vu}) \\ 0 & 0 & 1 \end{bmatrix} $$

is a coordinate transformation matrix, and

$$ \theta_{vu} = \pi - O_{vu} + B_{vu}. $$

In the equations above, B_{vu} and O_{vu} are the bearing angle and orientation angle respectively and d_{vu} is the distance between agent u and v measured by agent u. All these angles and distance can be measured locally by sensors. θ_{vu} is the relative heading of v from u.

In order to estimate the centroid, each robot maintains two pipelines, one for x coordinate and another for y coordinate of the estimated centroid. The initial input values for two pipelines are both zero,

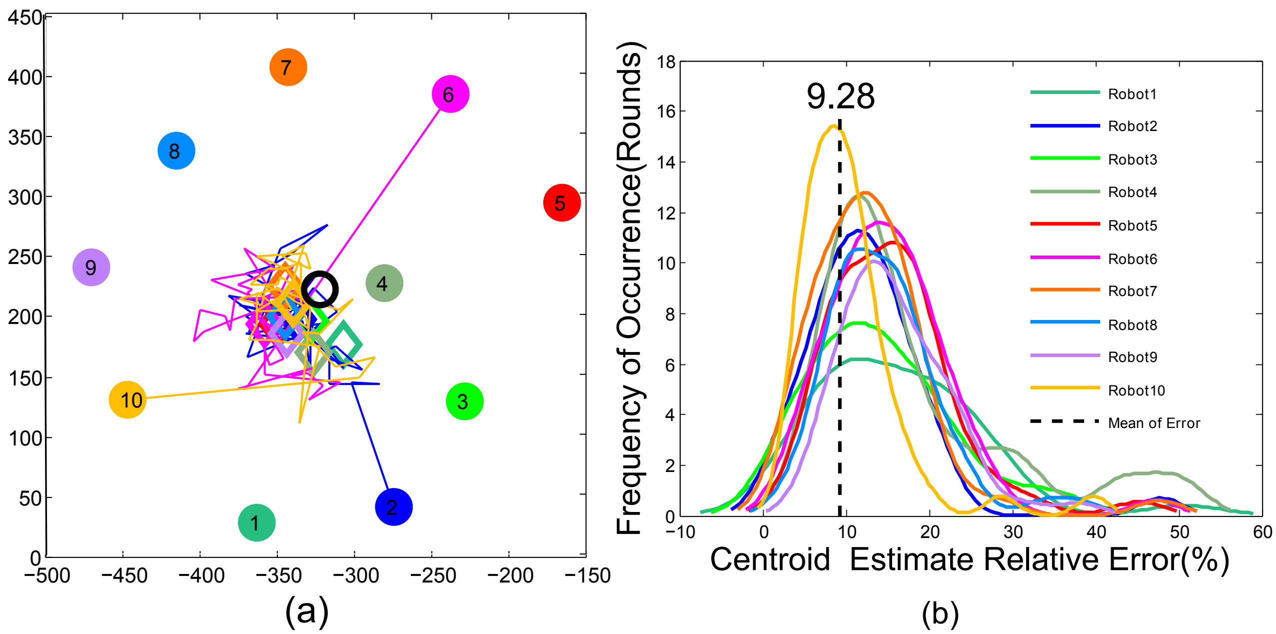


Figure 6: **(a)** Centroid estimation by 10 robots in a concave shape. The robots positions are designated by the solid circles numerically labeled. Three of robot's estimate over time is shown by the solid lines of similar color emanating from the circle. Each robots average estimate is shown by the colored diamonds. The true centroid is shown by the black circle. The average error is 9.28% for this experiment. **(b)** We fit Kernel density model to the distribution of centroid estimation error for each robot in the experiment illustrated in (a). The colors match with ones in the illustration(a). The mean error is 9.28% and is shown by dashed black line.

which represents the robot's position in its own reference frame. The relative poses of the neighbors are measured in every round. Robots use Algorithm 1 to do a pipelined consensus update each round. The robots transform each estimate in their neighbor's pipeline by using the aforementioned coordinate transformation. The values in the last cells of the x and y pipelines is considered the current estimation for the centroid.

4.2.1 *Physical Experiment, Ideal Conditions*

We tested pipelined centroid estimation on real agents. We used the r-one robot as our robotic agents. 10 robots were used in a configuration illustrated in Figure 6(a). This figure shows the robot's estimation of the centroid over time. In this experiment, the average error of the estimate to actual was 9%. The distribution of error for the centroid estimate is approximated by the kernel density estimation in Figure 6(b). In this experiment we used the AprilTag system [14]to measure distance between robots, eliminating error resulting from poor distance measurements on the robots. However, the error resulting from poor angular pose estimation using the infrared sensors still influenced the result. The resolution of the bearing and orientation is limited to 22.5° slices [10]. This result shows that the centroid is accurately estimated by pipelined consensus in the presence of sensor error.

4.2.2 *Simulated Experiment, Sensor Error*

In centroid estimation, sensor error is introduced by using the coordinate transformation. The angular and distance measurements sampled by the agent could be very noisy. In a normal linear average consensus, the estimate value evolves based upon a single sensor measurement or value taken at the beginning. This value has some unknown error from the sensor measurement that is never removed from the estimate. Our pipelined consensus algorithm continually re-samples the state of the network and sensors, and thus reduces error in estimation. In Figure 7, we demonstrate how the variance of the centroid estimation is related to the variance of the sensor measurements in a randomly generated unit disc graph. The distance and angle measurements are modeled by zero-mean Gaussian model. As we can see in the figure, the variance of the centroid estimation is almost linear to the variance of the sensor, and the angle measurement has a larger impact on the accuracy of the estimation. This is due to the coordinate transform, as angular errors will result in points moving a much greater distance from their actual positions than errors in the distance. Figure 8 provides a comparison between centroid estimation using linear average consensus and pipelined consensus. Linear average consensus has a larger estimation error, despite being much smoother than the values from pipelined consensus. Our pipelined consensus approach gives a much lower mean of relative error, but also has greater variance than the linear average consensus. Note that the linear average consensus does not reach a common value in this case because the sensor reading $\theta_{vu} \neq -\theta_{uv}$, $d_{vu} \neq d_{uv}$ due to noise. This results in the inconsistency when robots exchange their estimations using coordinate transformation, which causes the estimations of different robots to diverge.

4.2.3 *Physical Experiment, Population Changes*

Another feature of pipelined consensus is that it is self-stabilizing in regards to changing population and topology. We examined the effect of population changes on a network of robots performing centroid estimation. We began the test with 5 robots, and added and subtracted robots from the population in different areas of the network over time. Figure 9 shows the error in the estimate over time with changes in population. We add and remove robots only after the estimate has stabilized to a sufficient degree. The error in the estimation remains around 9% for all population sizes and changes.

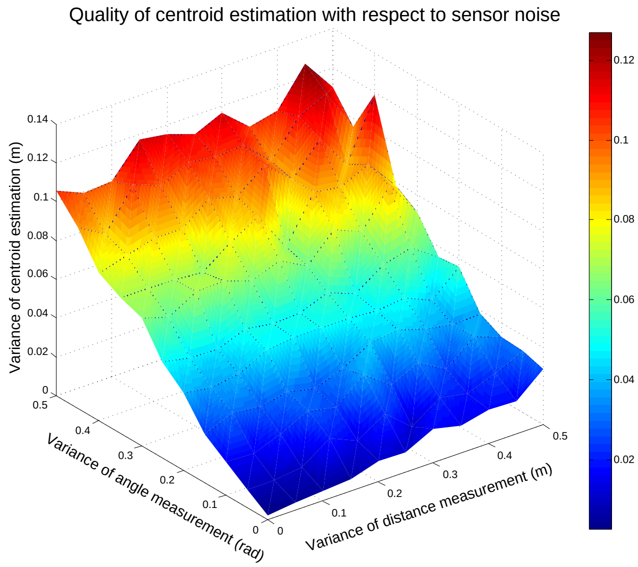


Figure 7: Simulation of centroid estimation with 20 robots. The graph is a randomly generated unit disc graph. The average degree is 7 while the network diameter is 3. We model the noisy sensor using zero-mean Gaussian model. The 3D plot here demonstrates the change of average variance of the centroid estimations of all robots as we increase variance of angle and distance measurements from 0 to 0.5.

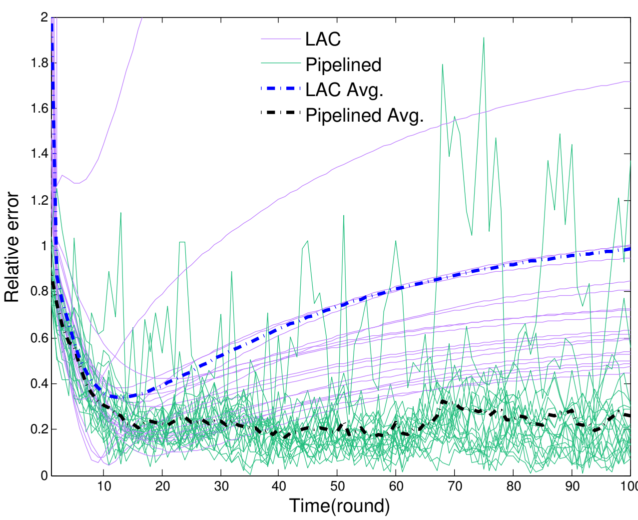


Figure 8: Comparison of normal linear average consensus and pipelined consensus over time on the same network as Figure 7. The y-axis shows the relative error, defined as the distance between the true and the estimated centroid, divided by the distance between the robot and the true centroid. Relative errors of all the robots using either pipelined or linear average consensus are plotted. The variances of both the distance and angle sensor are set to 0.1.

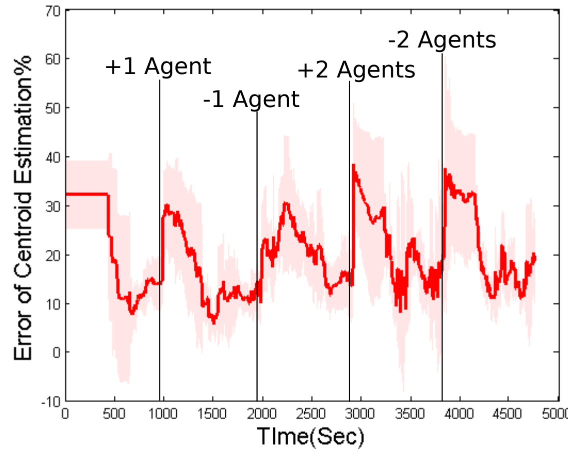


Figure 9: Illustration of robustness of pipelined consensus to population changes during the centroid estimation. Robots were added and removed over time after estimate stabilization. The robots removed and added were in different physical areas of the network each time. Each time a robot is added or removed from the network, error increases as the actual centroid no longer matches the estimated centroid. Over time, the estimate reconverges to the actual centroid within an error of 9% for all populations.

5. DISCUSSION AND LIMITATIONS

Our pipeline consensus algorithm achieves dynamic average tracking with high robustness to the initial condition, sensor errors, population changes, and communication failures. There are other algorithms that do the similar job, like the PI consensus filter designed in [3]. However, each agent must know the input signal for this approach. This may not be true in all multi-agent systems as some agents may not know the input signal for themselves or can only know the input signals for their neighbors. For example, in the centroid estimation task we described above, the only thing robots know about themselves is their position in their reference frame, which is always $[0,0]$. There is no knowledge about the input signal in this case, so the algorithm is not applicable.

Pipelined consensus is time-sensitive in nature. Data inserted in the pipeline will only be produced from the pipeline after *tau* rounds of successful consensus. However, the consensus operation is not related to time. The shifting values in pipeline consensus make it important to stay relatively synchronized. A poor configuration in the network or simple unluckiness may lead to an agent taking significantly longer time to produce an updated result. This can be seen in practice in Fig. 3. Some robots in the experiment produced an estimate much later than others, taking more time to achieve consensus τ times. This also can lead to stale data reentering the system and affecting the final result. For a system with a changing input signal, a robot who has not performed consensus for a relatively long time may perform consensus with another, more up-to-date robot. Performing consensus with the old data inserts error into the system, as the result for the temporally offset input values will be different.

The centroid estimation is sensitive to motion of the agents. This is because geometric measurements used in consensus are sensitive to the geometry of the network at the time the measurement was taken. The consensus estimate will become offset by the angular and translational motion. Since there is a lag of at least τ in pipeline consensus for information update, the error between the true centroid and the current estimation will grow the faster the agents change position. Hopefully, this error can be reduced by increasing the frequency of the communication.

6. CONCLUSION

We have demonstrated that Pipelined Consensus is a robust and practical extension to consensus algorithms for multi-agent systems. In all of our experiments, the algorithms handled many different types of errors well, quickly converging to accurate global estimates. In the future, we plan to rigorously study how to more accurately compute τ, which is related to features of the graph topology such as number of nodes, average degree, diameter, closeness, min-cut and so on. A mathematical characterization of τ will help ensure the effectiveness and efficiency of the pipelined consensus algorithm. Analysis of the variance introduced bu communications errors can help understand the convergence rates in tough communications environments. As for the centroid estimation, we plan to integrate it into a multi-robot manipulation task and test its performance when the robots actually move with the object.

7. ACKNOWLEDGEMENTS

This work has been supported by the National Science Foundation under CPS-1035716.

8. REFERENCES

[1] D. Angeli and P. Bliman. Convergence speed of distributed consensus and topology of the associated information spread. In *Decision and Control, 2007 46th IEEE Conference on*, pages 300–305. IEEE, 2007.

[2] R. Aragues, L. Carlone, C. Sagues, and G. Calafiore. Distributed centroid estimation from noisy relative measurements. *Systems & Control Letters*, 61(7):773–779, 2012.

[3] H. Bai, R. A. Freeman, and K. M. Lynch. Robust dynamic average consensus of time-varying inputs. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3104–3109. IEEE, 2010.

[4] M. Cao, A. S. Morse, and B. D. Anderson. Reaching a consensus in a dynamically changing environment: Convergence rates, measurement delays, and asynchronous events. *SIAM Journal on Control and Optimization*, 47(2):601–623, 2008.

[5] M. Franceschelli and A. Gasparri. Gossip-based centroid and common reference frame estimation in multiagent systems. 2014.

[6] J. Ghaderi and R. Srikant. Opinion dynamics in social networks: A local interaction game with stubborn agents. In *American Control Conference (ACC), 2013*, pages 1982–1987, June 2013.

[7] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988–1001, 2003.

[8] B. J. Julian, M. Angermann, M. Schwager, and D. Rus. A scalable information theoretic approach to distributed robot coordination. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 5187–5194. IEEE, 2011.

[9] J. McLurkin. *Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems*. PhD thesis, MIT ,USA, 2008.

[10] J. McLurkin, A. McMullen, N. Robbins, A. Chou, W. Li, M. John, C. Licato, N. Okeke, J. Rykowski, S. Kim, G. Habibi, W. Xie, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, A. Becker, L. Langford, J. Hunt, A. Boone, and K. Koch. A Robot System Design for Low-Cost Multi-Robot Manipulation. *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[11] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, 2004.

[12] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, 2009.

[13] A. Olshevsky and J. N. Tsitsiklis. Degree fluctuations and the convergence time of consensus algorithms. *Automatic Control, IEEE Transactions on*, 58(10):2626–2631, 2013.

[14] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[15] M. Schwager, D. Rus, and J.-J. Slotine. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.

[16] F. Shaw, A. Chiu, and J. McLurkin. Agreement on stochastic multi-robot systems with communication failures. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 6095–6100, Oct 2010.

[17] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Distributed sensor fusion using dynamic consensus. In *IFAC World Congress*, 2005.

[18] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Dynamic consensus on mobile networks. In *IFAC world congress*. Prague Czech Republic, 2005.

[19] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *Automatic Control, IEEE Transactions on*, 52(5):863–868, 2007.

[20] Z. Wang and M. Schwager. Multi-robot manipulation without communication. In *Proc. of the International Symposium on Distributed Robotic Systems (DARS 14)*, November 2014.

[21] P. Yang, R. A. Freeman, and K. M. Lynch. Distributed cooperative active sensing using consensus filters. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 405–410. IEEE, 2007.