Experimental verification of a scalable protocol for vehicle platooning

Thijs van Oorschot
Department of Dynamics & Control
Eindhoven University of Technology
5600 MB Eindhoven, The Netherlands
Email: t.j.a.v.oorschot@student.tue.nl

Mark Jeeninga
Department of Automatic Control
Lund University
SE-221 00 Lund, Sweden
Email: mark.jeeninga@control.lth.se

Emma Tegling
Department of Automatic Control
Lund University
SE-221 00 Lund, Sweden
Email: emma.tegling@control.lth.se

***Abstract*—This paper is concerned with a recently proposed scalable protocol for vehicle platooning, known as the *serial consensus* protocol. This achieves coordination of second-order integrator systems through a series connection of first-order conventional consensus protocols, and has been theoretically shown to have advantageous stability and performance properties. We implement this protocol on a vehicle platoon of five robots in a lab setting, where it is subject to measurement noise. We present experimental verification that the system is scalably stable, which is in accordance with theoretical findings. Moreover, we show theoretically that the parameters for which scalable stability occurs can be relaxed when information of the communication topology is available, which is also verified experimentally. In parallel, a conventional consensus protocol is implemented, which is known to *not* be scalably stable, which we also demonstrate in experiments. Lastly, we implement experiments to demonstrate the string stable behavior of the the serial consensus protocol in directed vehicle platoons, as predicted by theory, which again does not hold for conventional consensus.**

I. Introduction

The problem of vehicle platooning deals with the coordination of multiple vehicles using inter-vehicular measurements and/or limited communication. In its simplest form, it can be phrased as a mathematical control problem, where the goal is to design a control algorithm to follow a leader vehicle while maintaining a fixed inter-vehicular distance, or to achieve consensus among the behavior of all vehicles in the platoon. More abstractly, this problem corresponds to a *synchronization* or *consensus* problem, since we want to achieve consensus among the positions[1], velocities and accelerations of all vehicles in the platoon.

Synchronization, in turn, is a central topic in multi-agent systems which asks how and under which conditions a network of agents achieves consensus through distributed control laws. In addition, multi-agent systems typically feature a degree of locality, typically described by the network-topological constraints of the system. For synchronization problems, it is natural to formulate a control law based on relative feedback, which means that only measurements of the differences between states such as position, velocities and acceleration are available to the controller, for example implemented through radar measurements between neighbors. Jensen and Bamieh [4] give a thorough exposition on the topics of locality and relative feedback.

In this work, we are interested in distributed protocols for vehicle formations, subject to the locality constraints imposed by the string topology that is natural in traffic situations or platooning. Such protocols are known to feature unfavorable dynamic behaviors when the network grows large, including the well researched problem of string instability [7, 5] as well as *scale fragilities* [8] which imply that closed-loop stability is lost if the platoon grows beyond a certain size—e.g. see [6].

A recent paper by Hansson and Tegling [2] introduces a novel consensus protocol, referred to as the *serial consensus* protocol, and mathematically proves that the protocol results in a platoon that is both scalably stable and string stable [3]. This is achieved by a closed-loop design that replaces the conventional second-order consensus protocol by two first-order ones connected in series, and comes at the cost of (at most) one additional inter-vehicle communication step. The purpose of the current paper is to provide experimental verification of these results, and to compare its performance and robustness with a conventional approach. In addition, we

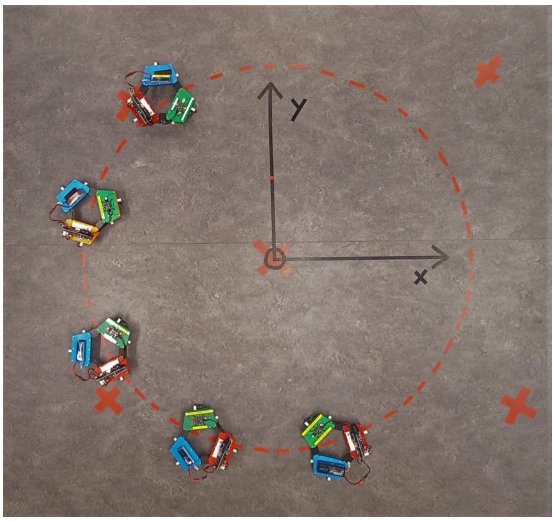

Fig. 1. A platoon of omnibots travelling along the circular trajectory.

[1]More precisely, the position is with respect to some constant inter-vehicular distance between neighboring vehicles. Synchronization of the exact position means that vehicles collide, which is naturally undesirable.

present a relaxation of the serial consensus algorithm, for which we mathematically prove and experimentally verify its scalable stability. It is demonstrated that this relaxation, however, leads to a loss of string stability.

The serial consensus protocol discussed in [2] only describes the behavior along the road that the platoon is following, and only applies to longitudinal control. To verify the protocol in a lab setting, we have implemented both protocols on vehicles that follow a circular trajectory. Given the limited range of our sensory setup, this means that long-duration experiments may be performed. It is noted that our experimentation setup is subject to measurement noise and features a discrete-time implementation of the protocols, both of which were not covered by the theory in [2]. We therefore argue that this provides a realistic implementation and verification of the protocol.

The remainder of this paper is structured as follows. In Section II we introduce the vehicle model and describe both the serial consensus protocol and a conventional consensus protocol for vehicle platooning. Section III presents the experimentation setup that is pursued in the paper. In Section IV we present the results of the experiments, and discuss to what extent our results support the theory developed in [2, 3]. The paper is concluded in Section V.

Notation

The time derivative of a signal $x(t)$ is denoted by $\dot{x}(t)$. We let $\mathbf{1}_k$ denote the all-ones vector of dimension k. We define the clipping function $\mathrm{clip}(x; a, b)$ for scalars $a < b$ as

$$\mathrm{clip}(x; a, b) := \begin{cases} a & \text{if } x \leq a \\ x & \text{if } a < x < b \\ b & \text{if } b \geq x \end{cases}.$$

We define the wrapping function $\mathrm{wrap}(x; r)$ as

$$\mathrm{wrap}(x; r) := \begin{cases} r(x + 2\pi) & \text{if } x < -\pi \\ rx & \text{if } -\pi \leq x \leq \pi \\ r(x + 2\pi) & \text{if } b > \pi \end{cases}.$$

II. Preliminaries and theoretical background

A. Vehicle model

Throughout this paper we consider a platoon of N vehicles. The vehicles are numbered such that vehicle $i = 1$ is the first vehicle in the platoon, and vehicle $i = N$ is the last vehicle. We assume that all vehicles in the platoon adhere to the same trajectory, and our goal is to control their position along the trajectory. For each vehicle i, we model this position by the scalar x_i, as depicted in Figure 2. Our aim is to control the vehicles in the platoon such that their inter-vehicular distance converges to a fixed value Δs^*. That is, we want to design controllers such that the positions x_i converge to

$$x_i = x_{i+1} + \Delta s^* \quad \text{for } i = 1, \ldots, N-1. \tag{1}$$

By defining the change of coordinates

$$\hat{x}_i(t) := x_i(t) - (i-1)\Delta s^*$$

we observe that (1) is equivalent to

$$\hat{x}_i(t) = \hat{x}_{i+1}(t) \quad \text{for } i = 1, \ldots, N-1. \tag{2}$$

We note from (2) that converging to a fixed inter-vehicular distance corresponds to achieving consensus among all vehicles in the coordinates x_i. Hence, vehicle platooning may be regarded as a consensus or synchronization problem.

We continue to model the dynamics of the vehicles in terms of $x_i(t)$, and collect these shifted positions in the vector $x \in \mathbb{R}^N$. Since Δs^* is constant, we have that $\dot{x}(t)$ represents the velocity of each vehicle along the trajectory. Similarly $\ddot{x}(t)$ represent the acceleration of the vehicles. We consider the very simplified model where each vehicle i is a point mass, and assume it is controlled by assigning the acceleration. Hence, we have that

$$\ddot{x}_i(t) = u_i(t). \tag{3}$$

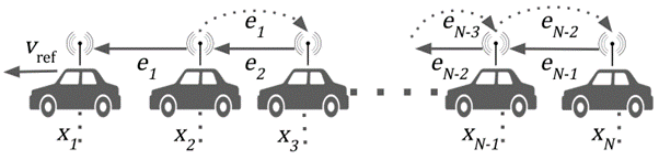

Fig. 2. Visualization of additional communication of relative position of neighbour's neighbour required to implement serial consensus.

B. Network model and definitions

The vehicles in the platoon can communicate according to some communication topology. We let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a directed graph that encodes the communication topology, where $\mathcal{V}$ with $|\mathcal{V}| = N$ denotes the set of vehicles and $\mathcal{E}$ denote the edges in the graph. The graph can be equivalently represented by the communication adjacency matrix $W \in \mathbb{R}^{N \times N}$, where $W_{i,j} = 1 \Leftrightarrow (j, i) \in \mathcal{E}$, and $W_{i,j} = 0$ otherwise. The graph is called undirected if $W^T = W$. The graph contains a connected spanning tree if for some $i \in \mathcal{V}$ there is a path from i to any other vertex $j \in \mathcal{V}$. Let $\mathcal{N}_i$ denote the set of neighbours of vehicle i. For a a *weighted* directed graph, we associate a positive weight $w_{i,j} > 0$ to each edge $(i, j) \in \mathcal{E}$. The associated weighted graph Laplacian is defined as

$$L_{i,j} = \begin{cases} -w_{i,j}, & \text{if } i \neq j \\ \sum_{k \neq i} w_{i,k}, & \text{if } i = j \end{cases}. \tag{4}$$

A graph contains a connected spanning tree if and only if the corresponding graph Laplacian L has a simple and unique eigenvalue at 0 and the remaining eigenvalues lie strictly in the right half plane (open RHP). Following the approach in [2, 3, 8], in this work we consider networks with a growing number of vehicles, to test the scalability properties of the controller. To this end, we define the *family* of communication graphs $\{\mathcal{G}_N\}$, where N is the size of the growing network.

C. Control design

We consider a linear second order state feedback controller, which can be written as

$$u(t) = -A_0\hat{x}(t) - A_1\dot{\hat{x}}(t) + u_{\text{ref}}(t), \tag{5}$$

where $u_{\text{ref}}(t) \in \mathbb{R}^N$ is an arbitrary control signal, and $A_k \in \mathbb{R}^{N\times N}$ defines the feedback on the k^{th} derivative of the vehicle shifted positions $\hat{x}$. The class of controllers that will be considered is restricted in three ways.

(i) The controller can only use relative feedback, i.e. $A_i\mathbf{1}_N = 0$ for $i = 0, 1$. Put differently, only the inter-vehicular measurements $\hat{x}_i - \hat{x}_j$ and $\dot{\hat{x}}_i - \dot{\hat{x}}_j$ are available to the controller.

(ii) The controller must have a limited gain, i.e. $||A_i||_\infty \leq c < \infty$ for $i = 0, 1$.

(iii) To limit the amount of communication between vehicles, the controller must only depend on the local 2-hop neighbourhood of each agent. This means that the controller only uses information that is at most 2 steps away in the communication network. Note that the zero-nonzero structure of A_0 and A_1 determines what information is communicated among the vehicles. Therefore, it must hold that $\sum_{k=0}^{2} W_{i,j}^k = 0 \Rightarrow [A_k]_{i,j} = 0$ for $k = 1, 2$.

Condition (iii) ensures that the serial consensus protocol [2] can be implemented, as this requires a 2-hop neighbourhood for second order agent dynamics. If a controller satisfies all three conditions, it is called a 2-step implementable relative feedback controller with respect to the adjacency matrix W. The family of all r-step implementable relative feedback controllers is given all $A_0, A_1 \in \mathbb{R}^{N\times N}$ that lie in the set

$$\mathcal{A}^r(W, c) := \left\{ A \;\middle|\; \begin{array}{c} \sum_{k=0}^{r} W_{i,j}^k = 0 \Rightarrow A_{i,j} = 0 \\ A\mathbf{1}_N = 0, ||A||_\infty \leq c \end{array} \right\}. \tag{6}$$

D. Scalable stability & string stability

For certain graph families, consensus algorithms in multi-agent systems where the agents have integrator dynamics of order two or higher are sensitive to scale fragilities, meaning that stability is lost as the network scales. To this end, we introduce the notion of scalable stability as defined in [8]. A consensus control design is scalably stable if the resulting closed loop system achieves consensus over any graph in the family $\{\mathcal{G}_N\}$, meaning that it is not subject to scale fragilities.

In addition, we introduce a notion of performance for agents with second order integrator dynamics. To this end, the relative position $e_p(t) := d + Lx(t)$ and velocity deviation $e_v(t) := \dot{x}(t) - v_{\text{ref}}\mathbf{1}_N$ are introduced, with $d \in \mathbb{R}^N$ being a vector of desired offsets and $v_{\text{ref}} \in \mathbb{R}$ being the desired vehicle velocity. In the context of vehicle platooning, these need to remain small to avoid vehicle collisions. The scalar e_v represents the deviation from the desired velocity, which needs to remain small to respect the speed limits of the vehicles. The multi-agent system described by (3) and feedback (5) is said to be string stable if there exists a fixed and finite α that ensures that

$$\sup_{t\geq 0} \left\| \begin{bmatrix} e_p(t) \\ e_v(t) \end{bmatrix} \right\|_\infty = \alpha \left\| \begin{bmatrix} e_p(0) \\ e_v(0) \end{bmatrix} \right\|_\infty. \tag{7}$$

A note on the difference between scalable stability and sting stability is in order. In the current context, scalable stability and string stability are two independent notions, since the latter does not require stability. Indeed, oscillatory trajectories of $e_p(t)$ and $e_v(t)$ can still satisfy (7), but do not correspond to (asymptotic) consensus. Instead, string stability is a performance notion on the transients of the consensus dynamics.

E. Conventional consensus

The conventional consensus algorithm that is considered in this paper is chosen so that the feedback gain matrices $A_{0,1}$ in (5) satisfy $A_0 = a_0 L$ and $A_1 = a_1 L$ for some graph Laplacian L, with $a_{0,1} > 0$ as feedback gains. The resulting input for an agent i is of the form

$$\begin{aligned} u_i(t) = -a_0 \sum_{j\in\mathcal{N}_i} w_{i,j}(\hat{x}_i(t) - \hat{x}_j(t)) \\ - a_1 \sum_{j\in\mathcal{N}_i} w_{i,j}(\dot{\hat{x}}_i(t) - \dot{\hat{x}}_j(t)) + u_{i,\text{ref}}(t) \end{aligned}$$

This results in a 1-step implementable relative feedback controller, as all $A_k = a_k L \in \mathcal{A}^1(W, c)$.

In [8] it is shown that no conventional consensus algorithm is scalably stable in directed ring graphs where the real part of one or more Laplacian eigenvalues approaches zero as N grows and at least one of these eigenvalues has a relatively large imaginary part. More specifically, the system becomes unstable when

$$a_1^2 \text{Re}\{\lambda_l\} \left[\left(\frac{\text{Re}\{\lambda_l\}}{\text{Im}\{\lambda_l\}} \right)^2 + 1 \right] - a_0 < 0 \tag{8}$$

for at least one $l \in 2, \ldots, N$. This is clearly satisfied for directed ring graphs, because $\text{Re}\{\lambda_l\} \to 0$ when $N \to \infty$ and $\text{Re}\{\lambda_l\} \neq 0$. Furthermore, conventional consensus exhibits poor dynamic behaviour in large scale networks. The transient error grows exponentially with the number of agents in a linear directed graphs. Therefore, conventional consensus does not meet the performance condition of string stability.

F. Serial consensus

The closed loop dynamics of the serial consensus in Laplace domain are defined as

$$(sI + L_1)(sI + L_2)\hat{X}(s) = \hat{U}(s). \tag{9}$$

We impose that both $L_{1,2}$ have the same underlying graph Laplacian multiplied by the positive gains $b_{1,2}$, i.e. that $L_1 = b_1 L$ and $L_2 = b_2 L$. By introducing the variables $\xi_1 = x$ and $\xi_2 = \dot{x} + b_1 L x$, the closed loop dynamics in state-space form can be stated as

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} -b_1 L & I \\ 0 & -b_2 L \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \begin{bmatrix} 0 \\ u_{\text{ref}} \end{bmatrix}. \tag{10}$$

The control input corresponding to this closed loop is given by (5) where $A_0 = b_1 b_2 L^2$ and $A_1 = (b_1 + b_2)L$. For a more intuitive notation, we introduce $\hat{a}_0 = b_1 b_2$ and

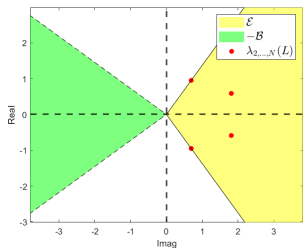

Fig. 3. Scissor condition for directed circular network with 5 vehicles.

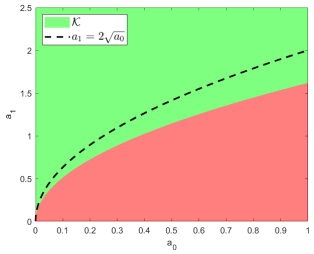

Fig. 4. Scissor condition for directed circular network with 5 vehicles visualized in $(\hat{a}_0, \hat{a}_1)$.

$\hat{a}_1 = b_1 + b_2$, allowing the control input to be written as (5) where $A_0 = \hat{a}_0 L^2$ and $A_1 = \hat{a}_1 L$, with $\hat{a}_{0,1} > 0$ as feedback gains. This results in a 2-step implementable relative feedback controller, as $A_0 = \hat{a}_0 L^2 \in \mathcal{A}^2(W, c)$. To implement this controller, additional information is required to be transmitted over the network. More specifically, this means that each agent requires the relative position between the neighbours of all its neighbours. This is visualized for a vehicle platoon in Figure 2. The control law for each vehicle i is given by

$$u_i(t) = -\hat{a}_0 \sum_{j\in\mathcal{N}_i} w_{i,j}\Bigg(\sum_{k\in\mathcal{N}_i} w_{i,k}(\hat{x}_i(t) - \hat{x}_k(t)) - \sum_{l\in\mathcal{N}_j} w_{j,l}(\hat{x}_j(t) - \hat{x}_l(t))\Bigg) - \hat{a}_1 \sum_{j\in\mathcal{N}_i} w_{i,j}(\dot{\hat{x}}_i(t) - \dot{\hat{x}}_j(t)) + u_{i,\text{ref}}(t)$$

Serial consensus has superior dynamic behaviour for large scale networks compared to conventional consensus. In [2] it is shown to be scalably stable for all graph families if $\hat{a}_1 \geq 2\sqrt{\hat{a}_0}$ (this is equivalent to imposing that $b_{1,2}$ are real valued). Here, we generalize this condition by saying that serial consensus is scalably stable if and only if the feedback gains $\hat{a}_{0,1}$ satisfy

$$\hat{a}_1 > \frac{2}{\sqrt{(\frac{1}{m_\lambda})^2 + 1}}\sqrt{\hat{a}_0}, \tag{11}$$

where

$$m_\lambda = \min\left\{ x \;\middle|\; \left|\frac{\text{Im}(\lambda_l)}{\text{Re}(\lambda_l)}\right| \leq x \quad \forall\, l = 2, 3, \dots, N \right\} \tag{12}$$

defines the minimum slope of the closed conic set $\mathcal{E}$ which still ensures that it contains all eigenvalues of the graph Laplacian L (this is equivalent to imposing that $b_{1,2}$ must be complex numbers within a conic set $\mathcal{B}$ in the complex plane), see Figure 3. The set $\mathcal{B}$ in terms of the feedback gains $(\hat{a}_1, \hat{a}_1)$ is denoted by $\mathcal{K}$, see Figure 4. Here, we use the information of the communication topology (more specifically, information of the eigenvalues of the Laplacian) to relax the stability condition. We term this relaxed condition the *scissor condition* as to the relation between set $\mathcal{E}$ and $\mathcal{B}$ resembles a scissor mechanism.

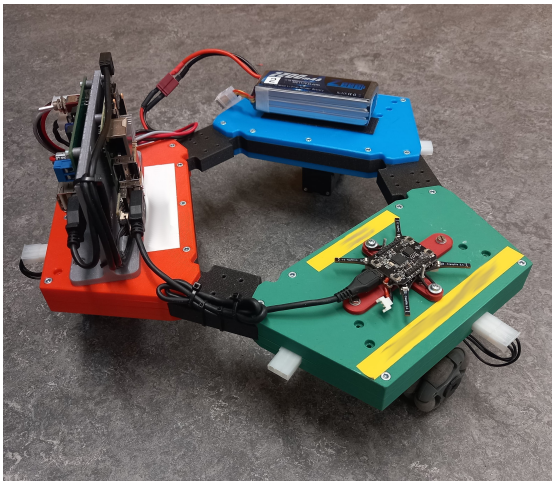

Fig. 5. Omnibot used in the experimentation setup.

Furthermore, [3] shows that serial consensus also achieves scalable performance through string stability. More specifically, when $\hat{a}_1 > 2\sqrt{\hat{a}_0}$, the worst-case behaviour gets bounded by the initial maximum deviation according to (7) with

$$\alpha = \frac{1}{\sqrt{\hat{a}_1^2 - 4\hat{a}_0}}(\hat{a}_1 + 2\max\{1, \hat{a}_0\}). \tag{13}$$

III. Experimentation Setup

To compare serial consensus to conventional consensus, both are implemented on robotic vehicles in an experimentation setup. The setup mimics vehicle platooning, as the agents move over a path along which they have to achieve consensus in cruising velocity. For practical reasons, the path is chosen to be a circle with radius r^*. The vehicles or agents are represented by omnibots[2] (see Figure 5). The setup should model each agent as a 2^{nd} order integrator, where the input is its acceleration along the circle and the states are its velocity and travelled distance along the circular path. This requires additional low-level control, as we should make sure that each agent stays on the circle. We want to decouple the control, in the sense that any deviation of an omnibot from the circle does not influence its dynamics along the path. Figure 1 shows the omnibots travelling over the path. A playlist containing video recordings of several experiments can be found in [1].

A. Decoupling control by change of coordinates

The kinematic design of the omnibots allows for decoupled control of its velocity in (x, y, θ)-coordinates in the Cartesian plane. This gives the Cartesian dynamics of an arbitrary omnibot i as

$$\begin{cases} \dot{x}_i = v_{x,i} \\ \dot{y}_i = v_{y,i} \\ \dot{\theta}_i = \omega_i. \end{cases} \tag{14}$$

[2]Omnibots are mobile robots equipped with three omniwheels, allowing for decoupled control of its velocity.

with $(v_{x,i}, v_{y,i}, \omega)$ representing the Cartesian velocity inputs. As the agents follow a circular path, a polar coordinate frame (s_i, r_i, θ_i) is introduced. Here, $s_i = r^*\phi_i \in [0, 2\pi r^*]$represents the travelled distance of agent i along the circle where $\phi_i = \mathrm{mod}\,(\arctan(y_i, x_i), 2\pi)$. $r_i = \sqrt{x_i^2 + y_i^2}$ represents the distance of agent i w.r.t. the center of the circle. The polar velocities in tangential direction $v_{t,i}$ and radial direction $v_{r,i}$ are introduced. These polar velocities are related to the Cartesian velocities through the rotation matrix $R(s_i)$ according to

$$\begin{bmatrix} v_{x,i} \\ v_{y,i} \\ \omega_i \end{bmatrix} = R(s_i) \begin{bmatrix} v_{r,i} \\ v_{t,i} \\ \omega_i \end{bmatrix}, \quad R(s_i) = \begin{bmatrix} \cos(\frac{s_i}{r^*}) & -\sin(\frac{s_i}{r^*}) & 0 \\ \sin(\frac{s_i}{r^*}) & \cos(\frac{s_i}{r^*}) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

To completely decouple the dynamics in the polar coordinate frame, the new input $\hat{v}_{t,i}$ is defined as $\hat{v}_{t,i} = \frac{r_i}{r^*} v_{t,i}$. This compensates for agents moving slower along the arc of a circle when it is outside of this circle (or vice versa inside the circle). The resulting dynamics in polar coordinates are

$$\begin{cases} \dot{s}_i = \hat{v}_{t,i} \\ \dot{r}_i = v_{r,i} \\ \dot{\theta}_i = \omega_i \end{cases}, \tag{16}$$

which are fully decoupled. The new control inputs are $(\hat{v}_{t,i}, v_{r,i}, \omega_i)$. The radial velocity $v_{r,i}$ is used to keep each agent on the circle. The rotational velocity ω_i is used to control the orientation of each agent. The tangential velocity $\hat{v}_{t,i}$ is used in the consensus algorithm.

1) Lateral control: The lateral dynamics of each omnibot in (16) are given by $\dot{r}_i = v_{r,i}$. As this is a single integrator system, a proportional feedback controller of the form $v_{r,i} = \alpha_i(r_i - r^*)$ is sufficient to stabilize the system and drive r_i asymptotically to the reference r^*.

2) Rotational control: The rotational dynamics of each omnibot in (16) are given by $\dot{\theta}_i = \omega_i$. This is again a single integrator system, meaning that a proportional feedback controller of the form $\omega_i = -\beta_i(\theta_i - \theta_{i,ref})$ is stabilizing and asymptotically drives the system to the reference orientation $\theta_{i,ref}$.

B. Artificial integration

The dynamics along the path in (16) are given by $\dot{s}_i = \hat{v}_{t,i}$. This is a 1^{st}-order integrator system, whereas the agents should be 2^{nd}-order integrators. Therefore, the input signal (which is acceleration) is numerically integrated to obtain $\hat{v}_{t,i}$. This $\hat{v}_{t,i}$ is then applied as the physical velocity of the omnibot. As the controller is implemented in discrete time with a finite sampling frequency, the resulting system is very similar to a second order integrator, but not equivalent. To see this, consider the discrete time state space of a pure 2^{nd}-order integrator

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau + \frac{1}{2}u(k)\tau^2 \\ \hat{v}_t(k+1) &= \hat{v}_t(k) + u(k)\tau, \end{aligned} \tag{17}$$

where τ denotes the sampling time. These dynamics are compared to the discrete time dynamics of the physical system

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau \\ \hat{v}_t(k+1) &= \hat{v}_t(k) + u(k)\tau, \end{aligned} \tag{18}$$

showing that the update step for $s(k+1)$ does not directly take the control input $u(k)$ into account. Instead, it is only indirectly influenced by $u(k-1)$ through $\hat{v}_t(k)$. The error of the update step in position of the physical system w.r.t. a pure integrator is denoted by $E(k) = \frac{1}{2}u(k)\tau^2$. It is clear that the error decreases for small control inputs $u(k)$ and a small sampling time τ. We therefore assume that for sufficiently high sampling frequencies the agents can be modelled as second order integrators.

The omnibots have a maximum velocity along the circle $\hat{v}_{t,max}$. To account for this, the numerically integrated velocity $\hat{v}_t$ is saturated in software using the clipping operator by imposing that

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau \\ \hat{v}_t(k+1) &= \mathrm{clip}(\hat{v}_t(k) + u(k)\tau; -\hat{v}_{t,max}, \hat{v}_{t,max}). \end{aligned} \tag{19}$$

C. Position and velocity measurement

The omnibots are equipped with a deck that emits a signal. This signal is detected by four lighthouses that compute the Cartesian position of each omnibot, denoted by (x_i, y_i, θ_i). These Cartesian coordinates are transformed to the polar coordinates (s_i, r_i, θ_i). The relative positions are obtained by taking the difference in travelled distance along the circle. To make sure that the differences are always in the range $[-\pi, \pi]$, they are computed with the wrapping function as $\Delta s_{ij} = \mathrm{wrap}(s_i - s_j; r^*)$.

The relative velocities are obtained by taking the differences of the numerical signals $\hat{v}_t$, which are assumed to be sufficiently similar to the actual velocity. This eliminates the need for measuring the velocity or designing an observer for the velocity.

We also wish to remark that, though the lighthouse system gives absolute measurements, the controllers only use the relative positions and velocities. This emulates relative measurements that could have been obtained with e.g. lidar measurements had that equipment been available.

D. Trajectory tracking verification

To verify the claim that each omnibot follows the circle and can be modelled as a second-order integrator, an experiment is performed where each agent starts with a random initial position and orientation and a constant tangential velocity $\hat{v}_t = 0.1$ [m/s]. The radius of the circular path is $r^* = 1$ [m]. Both lateral and rotational feedback gains are $\alpha = \beta = 0.5$. The control algorithm operates at a sampling frequency of 2 [Hz]. Figure 6 visualizes the measured vehicle trajectories in the Cartesian plane, showing that each omnibot indeed converges towards the circle and follows the path. Figure 7 highlights the lateral tracking performance, showing that the lateral position r indeed converges to the reference r^* for all agents.

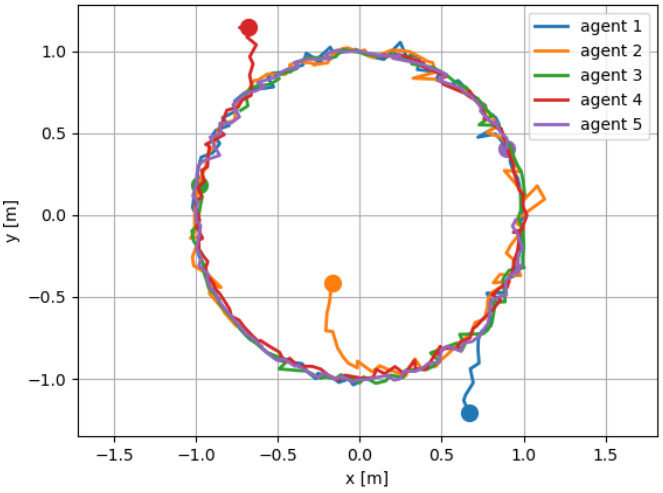

Fig. 6. Visualization of agent trajectories to verify the control structure.

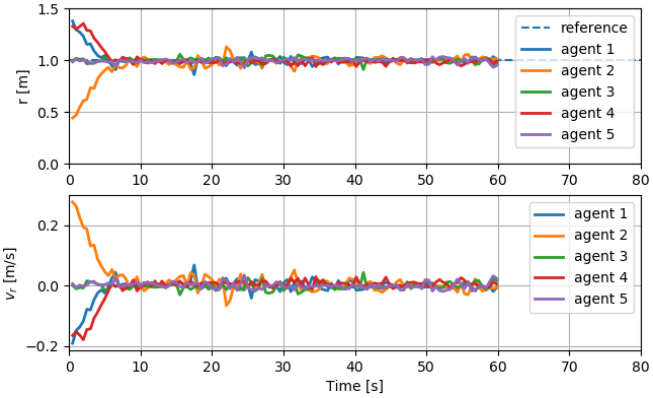

Fig. 7. Lateral tracking performance of control structure.

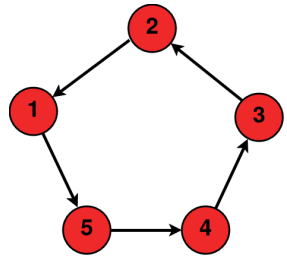

Fig. 8. Directed circular network

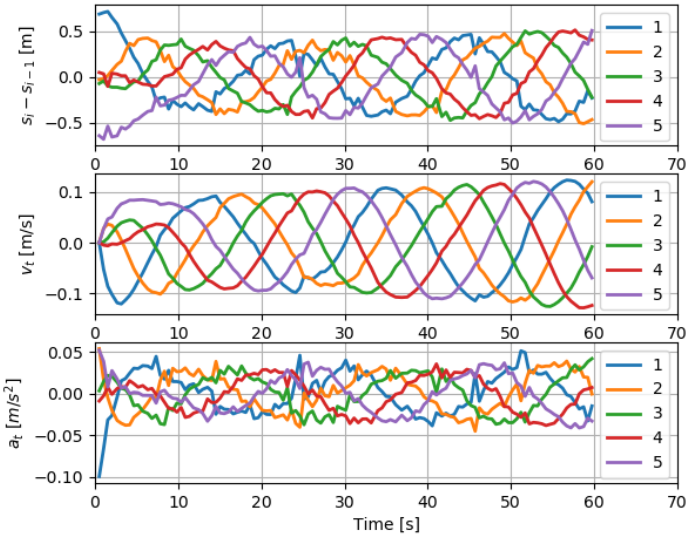

Fig. 9. Measured vehicle trajectories of experiment using serial consensus with feedback gains $\hat{a}_0 = 0.075$ and $\hat{a}_1 = 0.47$. The amplitude of the oscillation increases, and therefore we conclude that this combination of $(\hat{a}_0, \hat{a}_1)$ is unstable.

IV. Experimentation Results & Discussion

A. Scalable stability for circular graphs

First, an experiment is performed using a circular directed network with five agents (Figure 8) to verify the scalable stability. We measure the practical stability bound of the consensus algorithm ($a_{1,\text{practical}}$) and compare them to the theoretical bound ($a_{1,\text{theoretical}}$). We run the experiment for different values of a_0 and a_1, and we observe whether the system achieves consensus, giving us the practical stability bound. For example, Figure 9 shows the measured unstable response of serial consensus for a certain combination of feedback gains, since it is observed that the velocities steadily blow up. Figure 10 shows that the response is stable when $\hat{a}_1$ (i.e. the damping) is sufficiently increased, since the states remain neatly bounded.

1) Conventional consensus: The theoretical stability condition of conventional consensus in (8) for the circular network reads $a_1 > 0.9732\sqrt{a_0}$. This theoretical stability bound is verified experimentally by measuring the practical stability bound. Figure 11 visualizes the results of our experiments, showing that the practical stability bound is more conservative

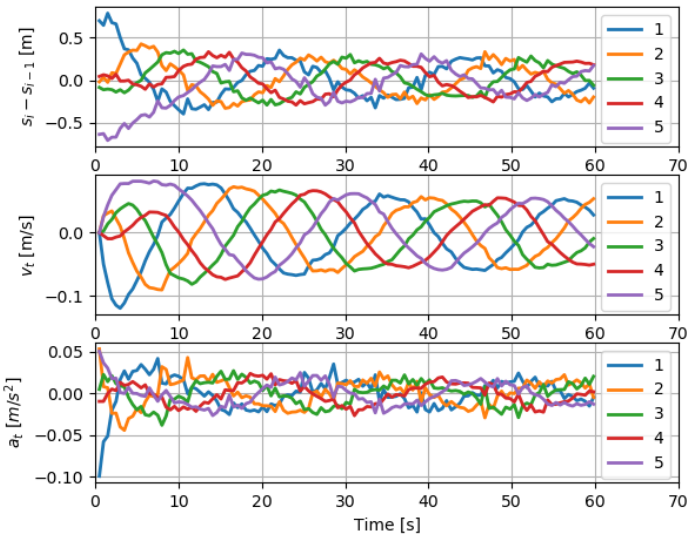

Fig. 10. Measured vehicle trajectories of experiment using serial consensus with feedback gains $\hat{a}_0 = 0.075$ and $\hat{a}_1 = 0.48$. The amplitude of the oscillation decreases, and therefore we conclude that this combination of $(\hat{a}_0, \hat{a}_1)$ is stable.

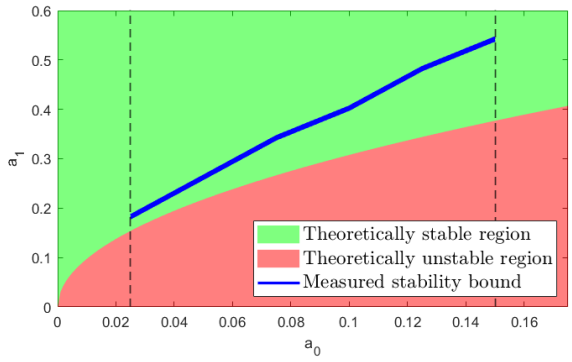

Fig. 11. Visualization of theoretical and practical stability bound of conventional consensus.

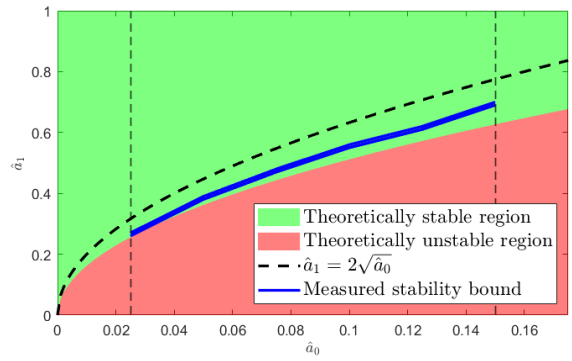

Fig. 12. Visualization of theoretical and practical stability bound of serial consensus.

in the sense that a higher a_1 is required to achieve consensus.

2) Serial consensus: The theoretical stability condition of serial consensus in (11) reads $\hat{a}_1 > 1.6180\sqrt{\hat{a}_0}$. This theoretical stability bound is verified experimentally by measuring the practical stability bound. Figure 12 visualizes the results, showing that the practical bound is again more conservative.

3) Discussion: The practical stability bound of both algorithms are more conservative than their theoretical bound, in the sense that a higher a_1 is required to achieve consensus. This makes sense when we realize that the feedback gains a_0 and a_1 can be the interpreted as the stiffness and damping of the system respectively. In practice, a higher damping is required to guarantee asymptotically stable consensus dynamics.

To visualize how much the unstable region is magnified, the ratio of the practical stability bound over the theoretical stability bound $\frac{a_{1,\text{practical}}}{a_{1,\text{theoretical}}}$ is visualized in Figure 13. This shows that the practical stability bound becomes more conservative for increasing a_0 and $\hat{a}_0$. Furthermore, it shows how the factor of conventional consensus is generally larger than the factor of serial consensus. This means that serial consensus is more robust to practical implementation than conventional consensus.

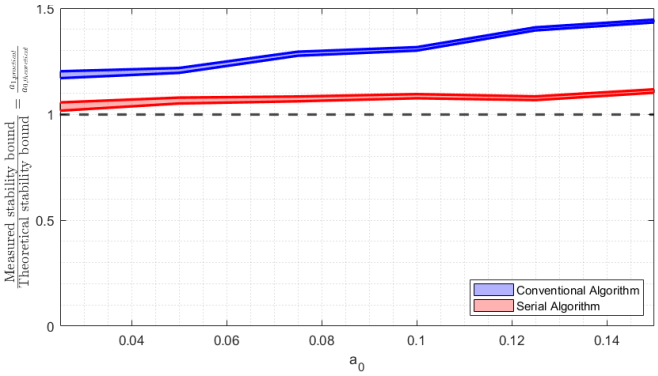

Fig. 13. Comparison between theoretical and practical stability bounds, both for conventional and serial consensus. The figure shows that serial consensus is more robust to practical implementation.

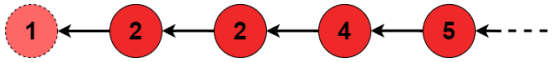

Fig. 14. Directed linear network

B. String stability for linear graphs

Next, an experiment is performed using a linear directed network (see Figure 14) to verify the string stability, that is, the property that disturbances do not propagate and increase along the string, but remain uniformly bounded. All agents start with consensus in position and a zero velocity, except the leading virtual agent. This agent has a constant (non-zero) reference velocity. Note that our experimentation setup features only five vehicles. By recording the trajectories of all predecessing vehicles, the experiments are repeated to artificially increase the length of the platoon to forty vehicles.

First, we demonstrate that in conventional consensus the transient error indeed grows exponentially with the number of agents. We then show that serial consensus is string stable in the sense that worst case behaviour is bounded, highlighting its superior scalable performance.

1) Conventional consensus: The results of the conventional consensus experiment with feedback gains $a_0 = 0.1$ and $a_1 = 0.6$ are visualized in Figure 15. The leading agent has a constant velocity of 0.05 [m/s], and the 40 following agents try to achieve the same velocity while maintaining the inter-vehicular distance.

The measurement noise of the sensors make it hard to see the exponential growth of the relative position. However, we do see the exponential growth in the overshoot of the following vehicles w.r.t. the leader reference velocity. Every next following vehicle has a bigger overshoot until the velocity saturation $\hat{v}_{t,max} = 0.18$ [m/s] is reached. The behaviour of conventional consensus indeed scales very poorly.

2) Serial consensus: The results of the serial consensus experiment with feedback gains $\hat{a}_0 = 0.1$ and $\hat{a}_1 = 0.8$ are visualized in Figure 16. The leading agent has a constant velocity of 0.10 [m/s], and the 30 following vehicles try to achieve the same velocity while maintaining the inter-vehicular distance. The results clearly show that now the relative position

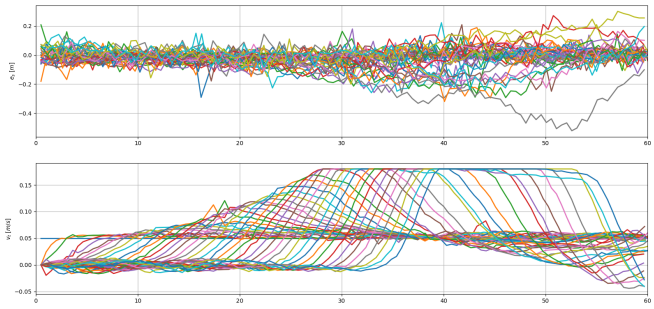

Fig. 15. Trajectories of vehicles in conventional consensus experiment showing the relative error e_p and velocity $\hat{v}_t$. Clearly this behaviour is not string stable.

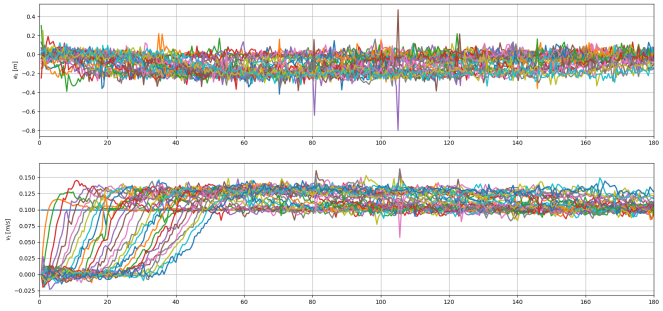

Fig. 16. Trajectories of vehicles in serial consensus experiment showing the relative error e_p and velocity $\hat{v}_t$. This behaviour is string stable as the transient error can be bounded independent of the number of agents.

and overshoot in velocity w.r.t. the leader agent do not grow unboundedly. Instead, the transient behaviour can be bounded by the initial maximum deviation according to (7), independent of the amount of vehicles in the platoon. This shows that the behaviour of serial consensus scales favourably.

3) Discussion: Comparing the transient errors between the two algorithms shows that serial consensus has a superior scalable performance to conventional consensus. Both relative position and the overshoot in velocity w.r.t. the leading vehicle do not grow unboundedly with the number of agents. In vehicle platooning, this is very important, as bounding the relative position error avoids vehicle collisions, and bounding the velocity overshoot ensures we respect the speed limits of the vehicles.

V. Conclusion

For this paper we have implemented a recent scalable protocol, known as the serial consensus protocol, in a lab setting with five robots. The implementation of the discretised protocol was subject to measurement noise. For the serial consensus protocol, it was mathematically proven in [2, 3] that it is scalably stable and string stable. In this paper we have presented experimental verification that these results hold in a real-life implementation. In parallel, a conventional consensus protocol was implemented, which was demonstrated to not be scalably stable and string stable, underlining the superiority of the serial consensus protocol. Moreover, we have introduced a relaxation of the serial consensus protocol, and demonstrated that scalable stability is still satisfied, but that the relaxation is subject to string instability.

Acknowledgments

The authors would like to thank Jonas Hansson for his valuable feedback and discussions.

References

[1] Serial consensus verification playlist. https://www.youtube.com/playlist?list=PLFCGpbpO3e3TsO2fvl4uOkMnRioEAE6_o.

[2] Jonas Hansson and Emma Tegling. A closed-loop design for scalable high-order consensus. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 7388–7394. IEEE, 2023.

[3] Jonas Hansson and Emma Tegling. Closed-loop design for scalable performance of vehicular formations. *arXiv preprint arXiv:2402.15208*, 2024.

[4] Emily Jensen and Bassam Bamieh. On structured-closed-loop versus structured-controller design: The case of relative measurement feedback. *arXiv preprint arXiv:2008.11291*, 2020.

[5] Pete Seiler, Aniruddha Pant, and Karl Hedrick. Disturbance propagation in vehicle strings. *IEEE Transactions on automatic control*, 49(10):1835–1842, 2004.

[6] Sonja Stüdli, María M Seron, and Richard H Middleton. Vehicular platoons in cyclic interconnections with constant inter-vehicle spacing. *IFAC-PapersOnLine*, 50(1):2511–2516, 2017.

[7] Darbha Swaroop and J Karl Hedrick. String stability of interconnected systems. *IEEE transactions on automatic control*, 41(3):349–357, 1996.

[8] Emma Tegling, Bassam Bamieh, and Henrik Sandberg. Scale fragilities in localized consensus dynamics. *Automatica*, 153:111046, 2023.