REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
17-10-2022	Final Report	1-Sep-2021 - 31-Aug-2022

4. TITLE AND SUBTITLE
Final Report: W911NF-17-S-0002: Platforms for Validation of Multi-Agent Autonomy

5a. CONTRACT NUMBER
W911NF-21-1-0350

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
111111

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Purdue University
Sponsored Program Services
155 S Grant Street
West Lafayette, IN 47907 -2114

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
79473-MI.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Shaoshuai Mou
UU	UU	UU	UU		19b. TELEPHONE NUMBER 765-494-6204

RPPR Final Report
as of 02-Nov-2022

Agency Code: 21XD

Proposal Number: 79473MI **Agreement Number: W911NF-21-1-0350**
INVESTIGATOR(S):

Name: Shaoshuai Mou
Email: mous@purdue.edu
Phone Number: 7654946204
Principal: Y

Name: Ph.D. Shreyas Sundaram
Email: sundara2@purdue.edu
Phone Number: 7654946204
Principal: N

Organization: **Purdue University**
Address: Sponsored Program Services, West Lafayette, IN 479072114
Country: USA

DUNS Number: 072051394 EIN: 356002041
Report Date: 30-Nov-2022 Date Received: 17-Oct-2022
Final Report for Period Beginning 01-Sep-2021 and Ending 31-Aug-2022
Title: W911NF-17-S-0002: Platforms for Validation of Multi-Agent Autonomy
Begin Performance Period: 01-Sep-2021 **End Performance Period:** 31-Aug-2022
Report Term: 0-Other

Submitted By: Shaoshuai Mou Email: mous@purdue.edu
Phone: (765) 494-6204

STEM Degrees: 2 **STEM Participants:** 2

Major Goals: Autonomous systems are quickly evolving from individual systems to multi-agent systems (MAS), i. e. a network of autonomous agents that work as a cohesive whole to accomplish complicated missions well beyond the capabilities of individual systems. While a significant amount of research has been devoted to development of theoretical algorithms for coordination in MAS, a key challenge is to test the proposed algorithms in realistic environments before deployment, especially those distributed algorithms for control, optimization and reinforcement learning in large-scale MAS. This has motivated the proposed research to provide platforms specially designed for validation of algorithms developed for MAS, which include a cloud-based simulator and a hardware-based testbed.

Accomplishments: We have devloped a versatile, easy-to-use simulation engine that can be leveraged by the swarm research community to rapidly develop, evaluate, and adjust swarm algorithms for a variety of common benchmark scenarios. This simulator is accessible at \url{ https://www.swarmsim.io/}. Our simulator is built on Microsoft AirSim and Unreal Engine, which provide support for vehicle and UAS models, together with a photorealistic graphics engine. We created an interface to allow swarm researchers to easily deploy and test their algorithms in complex simulated scenarios (such as search-and-rescue, ISR, and pursuit-evasion). This interface will allow researchers to choose the makeup of their swarm, and upload algorithms for different swarm tasks (such as patrolling, formation control, rendezvous, coverage, task allocation, etc.). The platform will maintain a library of benchmark scenarios, so that different algorithms can be easily and fairly evaluated against each other. We developed the simulator specifically to be deployed on the cloud, allowing ease of access to a wide class of researchers, obviating the need for expensive hardware and setup time, and allowing scaling to large swarms and environments.
We took initial steps to enable the ability to test reinforcement learning algorithms for various swarm tasks such as formation control.

We have also developed a multi-agent hardware testbed consisting of both unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV) to combine their respective advantages (such as the large payload capacity of UGVs, and the maneuverability and speed of UAVs). The testbed leverages both low-cost quadrotors (AR Drones, Crazyflies, Mambo) and advanced autonomous vehicles (Jackal UGV) available at the PI's lab, which can work collaboratively through local communications and coordination. The testbed supports modularized sensors to

enable more functionalities. Users can customize different sensors for a variety of tasks by simply plugging them into the expansion port of the deck. The testbed provides validations of distributed algorithms for formation control in the lab environment equipped with motion capture systems. A video can be found in \url{https://youtu.be/cy8Yzx5dmr0}

Training Opportunities: This project has helped to train two PhD students and two undergraduate students.

Results Dissemination: Nothing to Report

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: Nothing to Report

PARTICIPANTS:

Participant Type: Graduate Student (research assistant)
Participant: Tianyu Zhou
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Paulo Heredia
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Undergraduate Student
Participant: Tyler Fedrizi
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Undergraduate Student
Participant: Arpit Amin
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: PD/PI
Participant: Shaoshuai Mou
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 02-Nov-2022

Participant Type: PD/PI
Participant: Shreyas Sundaram
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Partners

,

I certify that the information in the report is complete and accurate:
Signature: Shaoshuai Mou
Signature Date: 10/17/22 10:13AM

Platforms for Validation of Multi-Agent Autonomy
(Final Report, October 17, 2022)

Shaoshuai Mou, Shreyas Sundaram

Center for Innovation in Control, Optimization and Networks (ICON) at Purdue University

1 Summary

This is the final report to our project "W911NF-17-S-0002: Platforms for Validation of Multi-Agent Autonomy" with funding from Army Research Office for 09/01/2021-08/31/2022. The report due date is 11/29/2022.

Autonomous systems are quickly evolving from individual systems to multi-agent systems (MAS), i.e. a network of autonomous agents that work as a cohesive whole to accomplish complicated missions well beyond the capabilities of individual systems. While a significant amount of research has been devoted to development of theoretical algorithms for coordination in MAS, a key challenge is to test the proposed algorithms in realistic environments before deployment, especially those distributed algorithms for control, optimization and reinforcement learning in large-scale MAS. This has motivated the **objective** of proposed research to provide platforms specially designed for validation of algorithms developed for MAS, which include a cloud-based simulator and a hardware-based testbed.

Finding 1 for Developing a Simulation-based Platform. We have developed a versatile, easy-to-use simulation engine that can be leveraged by the swarm research community to rapidly develop, evaluate, and adjust swarm algorithms for a variety of common benchmark scenarios. This simulator is accessible at `https://www.swarmsim.io/`. Our simulator is built on Microsoft AirSim and Unreal Engine, which provide support for vehicle and UAS models, together with a photorealistic graphics engine. We created an interface to allow swarm researchers to easily deploy and test their algorithms in complex simulated scenarios (such as search-and-rescue, ISR, and pursuit-evasion). This interface will allow researchers to choose the makeup of their swarm, and upload algorithms for different swarm tasks (such as patrolling, formation control, rendezvous, coverage, task allocation, etc.). The platform will maintain a library of benchmark scenarios, so that different algorithms can be easily and fairly evaluated against each other. We developed the simulator specifically to be deployed on the cloud, allowing ease of access to a wide class of researchers, obviating the need for expensive hardware and setup time, and allowing scaling to large swarms and environments. We took initial steps to enable the ability to test reinforcement learning algorithms for various swarm tasks such as formation control.

Finding 2 for Developing a Hardware-based Platform. We have also developed a multi-agent hardware testbed consisting of both unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV) to combine their respective advantages (such as the large payload capacity of UGVs, and the maneuverability and speed of UAVs). The testbed leverages both low-cost quadrotors (AR Drones, Crazyflies, Mambo) and advanced autonomous vehicles (Jackal UGV) available at the PI's lab, which can work collaboratively through local communications and coordination. The testbed supports modularized sensors to enable more functionalities. Users can customize different sensors for a variety of tasks by simply plugging them into the expansion port of the deck. The testbed provides validations of distributed algorithms for formation control in the lab environment equipped with motion capture systems. A video can be found in `https://youtu.be/cy8Yzx5dmr0`

2 Detailed Descriptions on the Cloud-based Simulator

Algorithm Description A substantial portion of our effort on the simulator was to facilitate implementation of distributed Linear Quadratic Regulator (LQR) control for swarm systems, including learning-based versions of such controllers. To describe the general setup, consider a linear time-invariant system

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \tag{1}$$

with a reference (target) state of the system, x_0 and reference state input u_0. The defined quadratic cost is given by two positive-definite cost matrices $\mathbf{Q}$ and $\mathbf{R}$ and defined by user. The defined cost-to-go for the infinite time solution is then given as:

$$\mathbf{J}(\mathbf{x}, \mathbf{u}) = \int_0^\infty \left(\tilde{\mathbf{x}}^\mathbf{T}\mathbf{Q}\tilde{\mathbf{x}} + \tilde{\mathbf{u}}^\mathbf{T}\mathbf{R}\tilde{\mathbf{u}}\right) dt \tag{2}$$

where $\tilde{\mathbf{x}} = x - x_0$ and $\tilde{\mathbf{u}} = u - u_0$. The optimal cost-to-go is where the quadratic form $\mathbf{J}^*(\mathbf{x}) = \mathbf{x}^\mathbf{T}\mathbf{P}\mathbf{x}$. Solving for P is done by utilizing the Continuous Algebraic Riccati Equation:

$$0 = \mathbf{P}\mathbf{A} + \mathbf{A}^\mathbf{T}\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-\mathbf{1}}\mathbf{B}^\mathbf{T}\mathbf{P} + \mathbf{Q} \tag{3}$$

Solving for the optimal feedback-policy is given as

$$\mathbf{u} = \mathbf{u}_0 + \mathbf{K}(\mathbf{x} - \mathbf{x_0}) \tag{4}$$

with K being defined as :

$$\mathbf{K} = -\mathbf{R}^{-\mathbf{1}}\mathbf{B}^\mathbf{T}\mathbf{P}. \tag{5}$$

To apply the above methdology to a quadrotor, the key assumption is that the body-rate controller is faster than the controlled state, so that the input can be bodyrates to the quadrotor system in addition to a collective normalized thrust constant. The total input vector is:

$$\mathbf{u} = [\omega_{des,x}, \omega_{des,y}, \omega_{des,z}, c_{des}] \tag{6}$$

The state of the quadrotor in this case is defined as position, orientation (represented as a quaternion), and velocity:

$$\mathbf{x} = [p_x, p_y, p_z, q_w, q_x, q_y, q_z, v_x, v_y, v_z] \tag{7}$$

We linearized the nonlinear dynamics of the quadrotor at its current state and utilized the state-dependent LQR [1] described above in the SWARMS simulator.

Simulation Setup In the SWARMS simulator, a formation control problem was implemented using a set of n agents where each agent was provided a minimum snap trajectory to follow using a cascaded PID position controller. In the formation control scenario provided in SWARMS, a set of waypoints are provided for a given swarm to follow. An equidistant formation is calculated around each waypoint for n agents. Therefore each agent will have a set of coordinates it needs to route to in order to stay in a formation. A trajectory is calculated for each agent using a minimum snap trajectory generator where time between each waypoint was manually selected in order to not exceed jerk and snap limits. This trajectory was tracked using the state dependent LQR described above.

There were several roadblocks along the way, and implementing the complete LQR is still underway. One of the biggest issues was timing when to send commands. The current approach was a temporal tracker solution. Since the trajectory was defined prior to execution, we knew what

the state of the drone should be at each setpoint. We can then compute the control input needed to drive the system through the whole trajectory and with taking into account the previous control input. However, due to usage of a local system with fewer compute resources and the nature of Microsoft AirSim to speed up or slow down the clock depending on compute resources, there was not a fixed timestep between each point. The trajectory generator computes each trajectory for each drone at a fixed timestep.

Another issue that was of importance is sending the command through the python API of Microsoft AirSim. The API is limited to commands at 50hz. In the implemented paper [1], they expect a rate command update of 100hz with a state update of 250 hz. By instantiating two python API clients each as its own process and queue, a state update higher than 250hz was achieved when running a single drone. However, using a queue for the commands was not sufficient.

Future progress will need to be made on understanding the timing the commands correctly, as well as validating the state dependent LQR with better testing. Since the multirotor physics models runs seperately of Unreal Engine, a forward thinking approach is rewrite it in python, and utilize that for fast debugging. This would validate all control based algorithms before running them in AirSim since rendering the engine on lower capacity machine influences the performance.

Simulation Setup The SWARM simulation platform (`https://swarmsim.io`) was utilized for scaling up the number of agents. The preliminary algorithms were implemented offline with a python framework and then inserted into SWARM for fast debugging and scaling for a higher number of agents than that would be possible for on a local computer.

Summary of Outcomes for Simulator Overall, the SWARM simulation platform has demonstrated its potential for allowing researchers to quickly test swarm algorithms in a cloud environment. The platform can be accessed and tested at `https://swarmsim.io`. An overview of the platform was presented by Tyler Fedrizzi (former student at Purdue and creator of hte platform) at the ICRA 2022 Workshop on "Releasing Robots into the Wild: Simulations, Benchmarks, and Deployment." Progress was also made on implementing LQR and learning-based controllers on the platform. However, there are additional challenges to be addressed to fully implement such lower-level control algorithms, due to technical issues pertaining to sampling rates for the underlying platform. The project has identified additional avenues for research and development to overcome these issues.

3 Detailed Description on the Hardware-based Testbed

Algorithm Description

Consider a linear system

$$\dot{x} = Ax + Bu \tag{8}$$

Where $x \in \mathbb{R}^n$ is the state of the system. $u \in \mathbb{R}^m$ is the control input. $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are unknown constant matrices. In order to continue with the computational method, the system is assumed to be stabilizable.

The optimal control gain K can be found by solving Algebraic Riccati Equation (ARE)

$$A^\top P + PA + Q - PBR^{-1}B^\top P = 0 \tag{9}$$

Equation (9) has a unique symmetric positive definite solution P^* and the optimal gain K^* can be found as

$$K^* = R^{-1}B^\top P^*. \tag{10}$$

With unknown dynamic, Algebraic Riccati Equation could be extremely hard to solve when we have unknown variables and large scales (multi-agents). Thus, reinforcement learning based method is involved (Adaptive Dynamic Programming).

Two operators are defined:

$$P \in \mathbb{R}^{n\times n} \to \hat{P} \in \mathbb{R}^{\frac{1}{2}n(n+1)}, \text{ and } x \in \mathbb{R}^n \to \bar{x} \in \mathbb{R}^{\frac{1}{2}n(n+1)} \tag{11}$$

where

$$\hat{P} = [p_{11}, 2p_{12}, \ldots, 2p_{1n}, p_{22}, 2p_{23}, \ldots, 2p_{n-1,n}, p_{nn}]^\top \tag{12}$$

$$\bar{x} = [x_1^2, x_1x_2, \ldots, x_1x_n, x_2^2, x_2x_3, \ldots, x_{n-1}x_n, x_n^2]^\top \tag{13}$$

By Kronecker product representation

$$x^\top Q_k x = (x^\top \otimes x^\top)\ \mathrm{vec}(Q_k) \tag{14}$$

and

$$(u + K_k x)^\top R K_{k+1} x = \left[(x^\top \otimes x^\top)(I_n \otimes K_k^\top R) + (x^\top \otimes u^\top)(I_n \otimes R)\right]\mathrm{vec}(K_{k+1}) \tag{15}$$

With an exploration time T and number of sample collection l, the time step between each point is $\frac{T}{l}$. For positive integer l, we define the matrices $\delta_{xx} \in \mathbb{R}^{l\times\frac{1}{2}n(n+1)}$, $I_{xx} \in \mathbb{R}^{l\times n^2}$, $I_{xu} \in \mathbb{R}^{l\times mn}$, such that

$$\begin{aligned}
\delta_{xx} &= [\bar{x}(t_1) - \bar{x}(t_0), \bar{x}(t_2) - \bar{x}(t_1), \ldots, \bar{x}(t_l) - \bar{x}_(t_{l-1})]^\top, \\
I_{xx} &= \left[\int_{t_0}^{t_1} x \otimes x d\tau, \int_{t_1}^{t_2} x \otimes x d\tau, \ldots, \int_{t_{l-1}}^{t_l} x \otimes x d\tau\right]^\top, \\
I_{xu} &= \left[\int_{t_0}^{t_1} x \otimes u d\tau, \int_{t_1}^{t_2} x \otimes u d\tau, \ldots, \int_{t_{l-1}}^{t_l} x \otimes u d\tau\right]^\top,
\end{aligned} \tag{16}$$

where $0 \le t_0 < t_1 < \ldots < t_l$. Equation (16) was computed by using the state and control input data collected in the exploration period.

For any stabilizing gain K_k,

$$\Theta_k \begin{bmatrix} \hat{P}_k \\ \mathrm{vec}(K_{k+1}) \end{bmatrix} = \Xi_k \tag{17}$$

where $\Theta_k \in \mathbb{R}^{l\times\frac{1}{2}n(n+1)+mn}$ and $\Xi_k \in \mathbb{R}^l$ can be computed using the variables calculated from Equation (16):

$$\Theta_k = [\delta_{xx}, -2I_{xx}(I_n \otimes K_k^\top R) - 2I_{xu}(I_n \otimes R)] \tag{18}$$

$$\Xi_k = -I_{xx}\mathrm{vec}(Q_k) \tag{19}$$

When Θ_k has full column rank, Equation (17) can be solved as

$$\begin{bmatrix} \hat{P}_k \\ \mathrm{vec}(K_{k+1}) \end{bmatrix} = (\Theta_k^\top \Theta_k)^{-1}\Theta_k^\top \Xi_k \tag{20}$$

Rank condition: If there exists an integer $l_0 > 0$, such that

$$\mathrm{rank}([I_{xx}, I_{xu}]) = \frac{n(n+1)}{2} + mn \tag{21}$$

$\forall l > l_0$, then Θ_k has full column rank for all $k \in \mathbb{Z}_+$.

Algorithm 1 Computational Adaptive Dynamic Programming

1: Initialize $k = 0$, K_0 is stabilizing
2: Employ $u = -K_0 x + e$ as the input on the time interval $[t_0, t_l]$, where e is the exploration noise.
3: Compute δ_{xx}, I_{xx}, and I_{xu} to meet the Equation (21) in terms of the rank condition.
4: **while** $||P_k - P_{k-1}|| \leq \varepsilon$ **do**
5: Solve P_k and K_{k+1} using Equation (20)
6: $k \leftarrow k + 1$
7: **end while**
8: Use $u = -K_k x$ as the approximated optimal control policy.

After the learning process, We further apply a linear transformation

$$z := Tx = \begin{pmatrix} x_2 - x_1 \\ x_3 - x_1 \\ \vdots \\ x_n - x_1 \\ \frac{1}{n}\sum_{i=1}^{n} x_i \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ \bar{z} \end{pmatrix} \tag{22}$$

where $z \in \mathbb{R}^{nN}$. This includes the relative position and velocity between agents and leader (formation control) and the centroid of all agents (target tracking). The desired states q is an array contains desired differences of states and the target state. By applying the optimal control gain to the system after linear transformation, the system will make desired formation at desired location

Codes `https://github.com/tianyuzhou-sam/Multi-agent-Formation-Control-Using-Reinforcement-Lear`
`git`

Experimental Setup The experiment is set up using three Mambo quadrotors and Jackal UGV. The state information is captured using nine Qualisys Motion Capture System (Mocap).There is a ground station to handle all the computation and send commends to Mambos. Because we cannot access to Mambo's lower controller, I built a simulator to run parallel with real quadrotors. The quadrotors resume the learning process in real world. Once the learning is done, the formation runs fully in real world . The relative states (such as relative position and velocity) between all following agents and the leading agents are defined by user. The center of the formation is set to Jackal's position. The experiment uses real-time feedback system. The real states of all agents are feed into the algorithm to compute the future input. By having this feature, the effect of turbulence is minimized and the quadrotors could keep Jackal in center even if Jackal is moving. A link to the demo video is: `https://youtu.be/cy8Yzx5dmr0`

References

[1] Philipp Foehn and Davide Scaramuzza. Onboard state dependent lqr for agile quadrotors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6566–6572, 2018.