IEEE VTS

Connecting the Mobile World

WEBINAR SERIES ON ADVANCED MOBILITY

IEEE VTS
Connecting the Mobile World

WEBINAR SERIES ON ADVANCED MOBILITY

Multi-Vehicle Mission Scenarios

01 April 2024

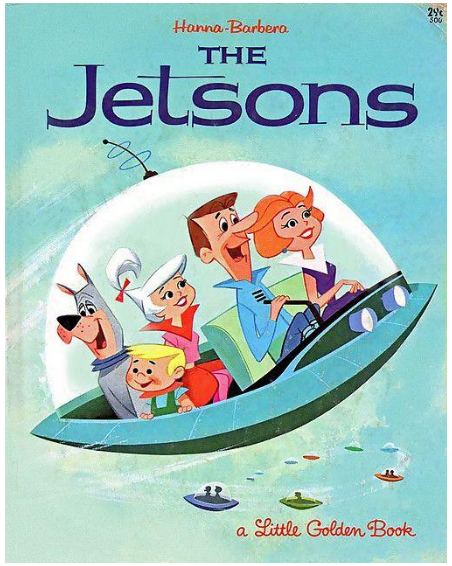
Dr. Marc Compere, Associate Professor
(comperem@erau.edu, https://faculty.erau.edu/Marc.Compere)
Department of Mechanical Engineering
Embry-Riddle Aeronautical University, Daytona Beach FL, USA

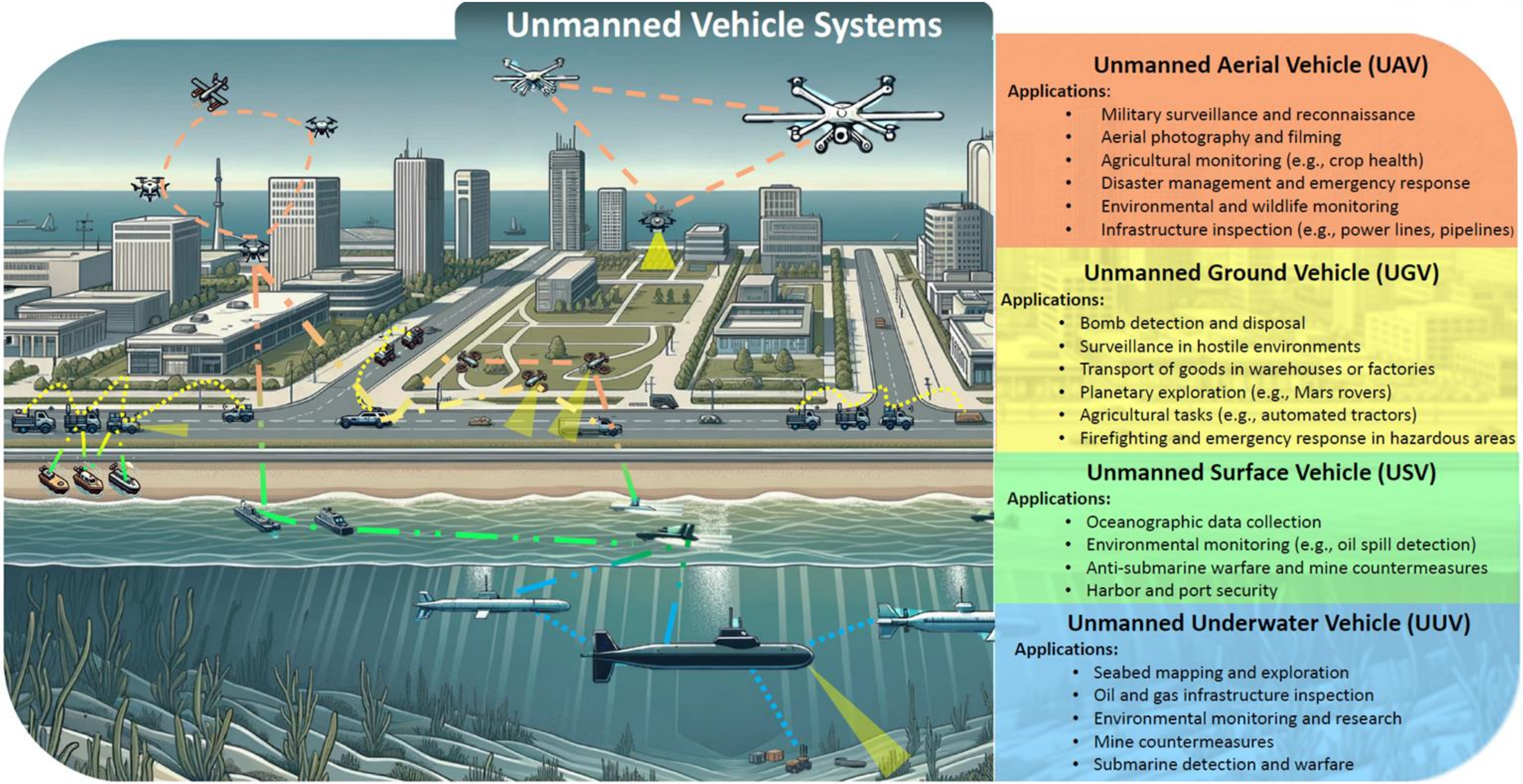
Agenda

1. Brief Intro and Background
2. The Mobility Virtual Environment
3. Multi-Vehicle Testing: Virtual and Real
    1. Real vehicle tests, outdoors, with references
    2. Simulation for the remainder of this presentation
4. MoVE architecture:
    1. High-level Diagram
    2. Individual Vehicle Models
    3. Priority-based Behavior Table
    4. Vehicle-to-Vehicle Communication Model (v2v)
    5. Mission Sequencer Table
5. Mission Rehearsal Example
6. The complete Mission Simulation Architecture
7. Documentation: Code, Docs, References

Motivation

It’s happening!





G. Liu et al., “Generative AI for Unmanned Vehicle swarms: challenges, applications and opportunities,” arXiv.org, Feb. 28, 2024. https://arxiv.org/abs/2402.18062

The Need

The UAS research community needs a way to rehearse multi-vehicle scenarios along a realism spectrum from simulation-only to all real vehicles with real people.

Ideally, the community needs:

- Common tools for easily sharing scenarios and data
- Open-source software available to many*

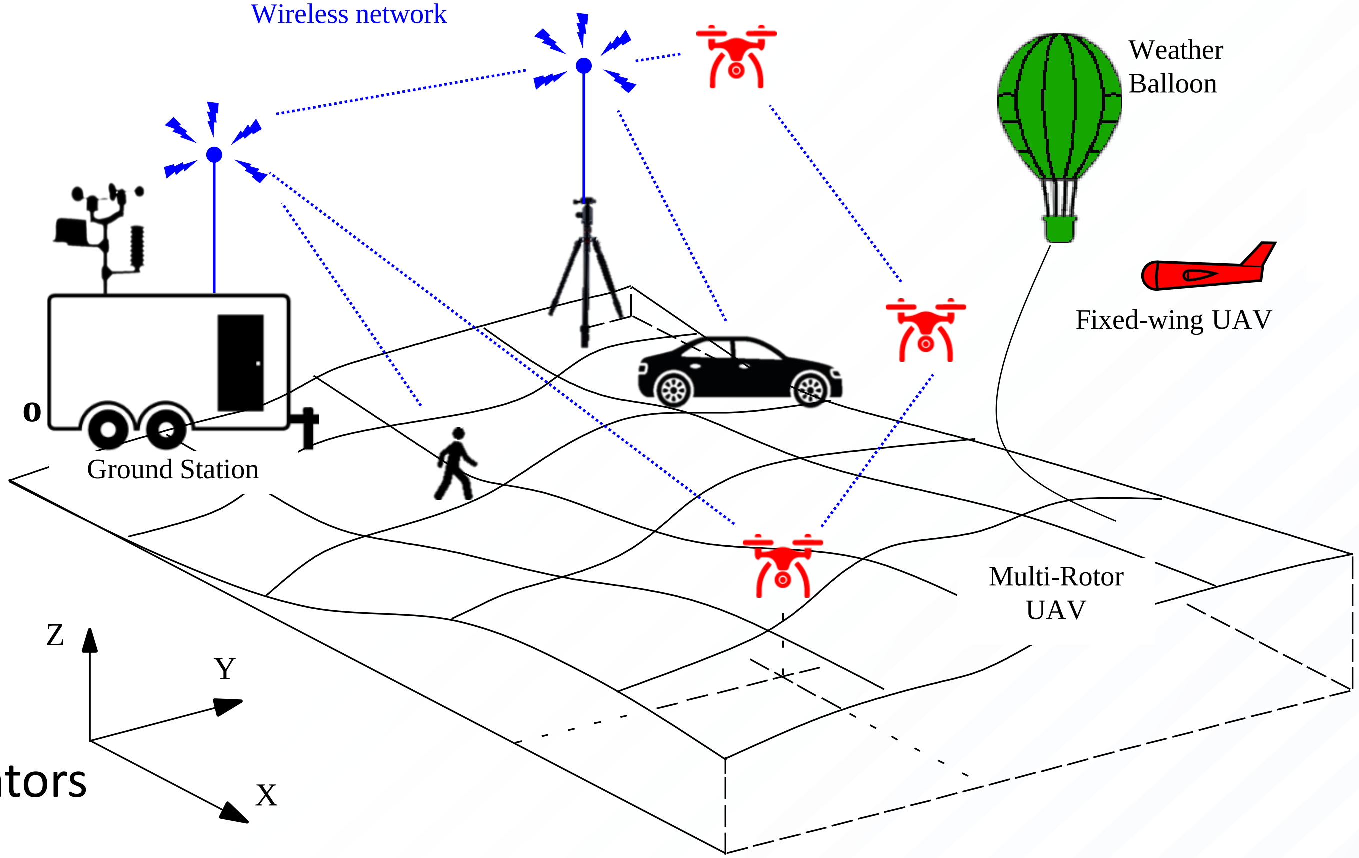
  note: *the US military has good open-source policy
- Simple software that Just Works with standard computers (no hardware or high-end graphics dependencies)
- A way to leverage common devices: iPhones, Androids, Raspberry Pi / Arduino
- Methods that capture wireless networks, including cellular networks

Introduce MoVE the Mobility Virtual Environment

MoVE: Mobility Virtual Environment

MoVE is a network-centric framework for testing real vehicles, virtual vehicles, and pedestrians in the same coordinate system with a common timestamp.

Real and Virtual pedestrians



Virtual vehicles, Repeatable traffic

Repeatable challenge cases

Real Vehicle, Real Operators

MoVE Vehicle Motion

There are three distinct ways of using **MoVE**

1. Simulation Only

Today's presentation!

2. Real Vehicles and Real Pedestrians
- Stream GPS locations over wireless network:
  - iPhone SensorLog app, Android HyperIMU app
- Motion in the real world is conveyed to the virtual environment

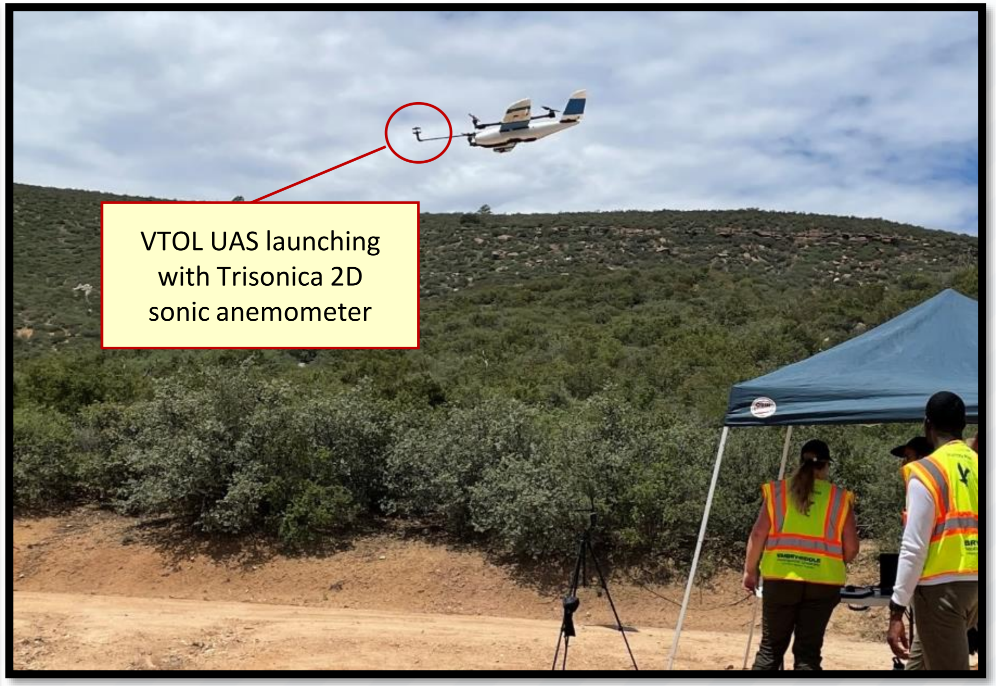
References slide has [1-6], but mainly

Compere, M. D., Adkins, K. A., Krishnan, A. M., Schroeder, R., & James, C. N. (2024). The mobility virtual environment (MoVE): an open source framework for gathering and visualizing atmospheric observations using multiple vehicle-based sensors. *Environmental Science: Atmospheres*, https://pubs.rsc.org/en/content/articlehtml/2024/ea/d2ea00106c

3. Mixed Real and Virtual:
- Virtual vehicles can avoid real people
- Real vehicles can see repeatable virtual challenges (with AR headset)

Field Campaign with Real Vehicles, Arizona, 2021

- MoVE was critical for gathering multiple vehicle locations and sensor data across 2 weeks of flights!



Compere, M. D., Adkins, K. A., Krishnan, A. M., Schroeder, R., & James, C. N. (2024). The mobility virtual environment (MoVE): an open source framework for gathering and visualizing atmospheric observations using multiple vehicle-based sensors. *Environmental Science: Atmospheres*, https://pubs.rsc.org/en/content/articlehtml/2024/ea/d2ea00106c

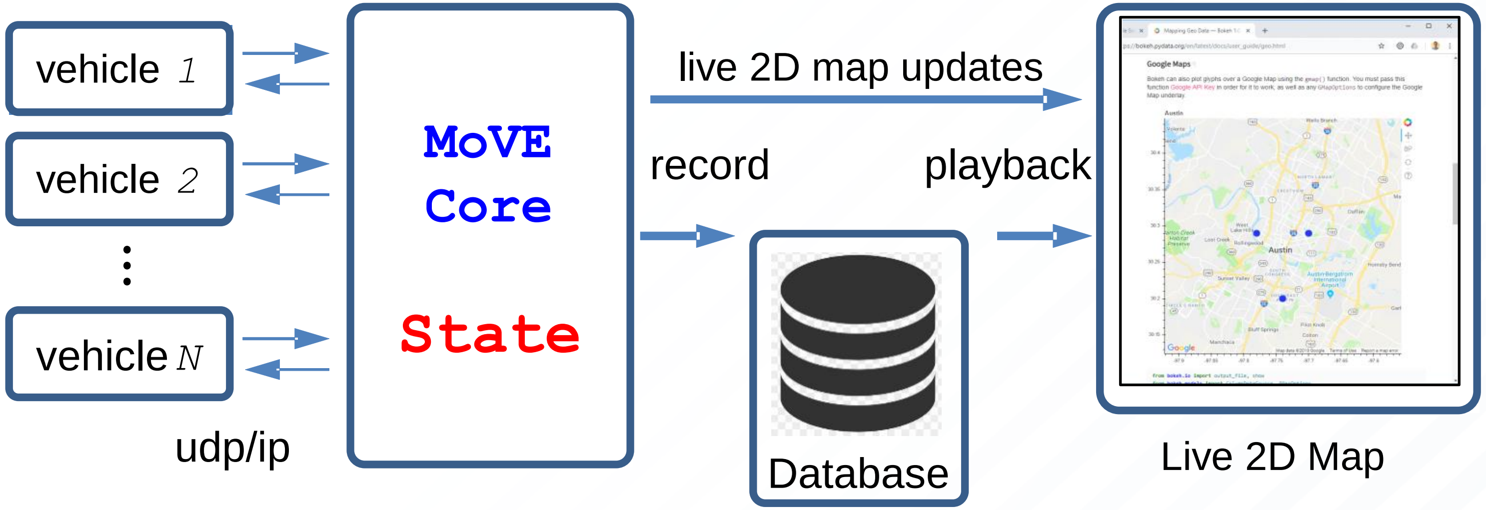
MoVE Multi-Vehicle Simulations

MoVE Software Architecture

How does a MoVE experiment work?

1. Read config file,
Launch processes

3. All vehicles report position and health status to `Core`



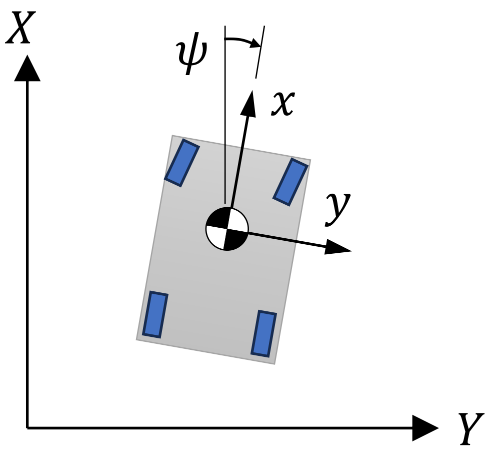
2. Issue `runState` commands:
`Ready, Set, Go!, Pause, Stop`

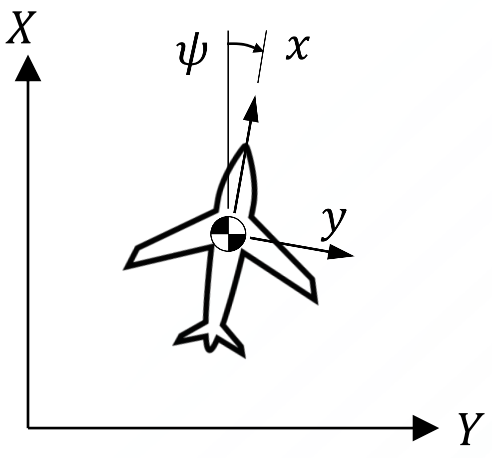
4. `Core` aggregates all positions into `State` and logs with timestamps

A Simple Kinematic Vehicle Model

MoVE's vehicle models have a characteristic length, L, but otherwise are similar to:

- Dubins Car [1]
- Dubins Airplane [2]





Kinematic equations of motion in the body-fixed xy frame are:

$$v_g^{xy} = \begin{bmatrix} v_x & \hat{\imath} \\ b\dot{\psi} & \hat{\jmath} \end{bmatrix} \qquad \dot{\psi} = \left(\frac{v_x}{L}\right)\delta_{steer}$$

Transform body-fixed velocities in xy frame to the inertial XY frame with:

$$v_g^{XY} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot v_g^{xy}$$

Then integrate XY velocities for XY positions:

$$\begin{bmatrix} X_g \\ Y_g \end{bmatrix} = \int v_g^{XY}\, dt \qquad \psi = \int \dot{\psi}\, dt$$

[1] https://en.wikipedia.org/wiki/Dubins_path
[2] https://ieeexplore.ieee.org/document/4434966

Priority Based Behaviors

- Rodney Brooks in the late 1980's introduced the concept of fast, reactive behaviors based on sensor inputs [1].
- Behaviors are separate threads within a vehicle process
- Behaviors with highest priority determine vehicle motion

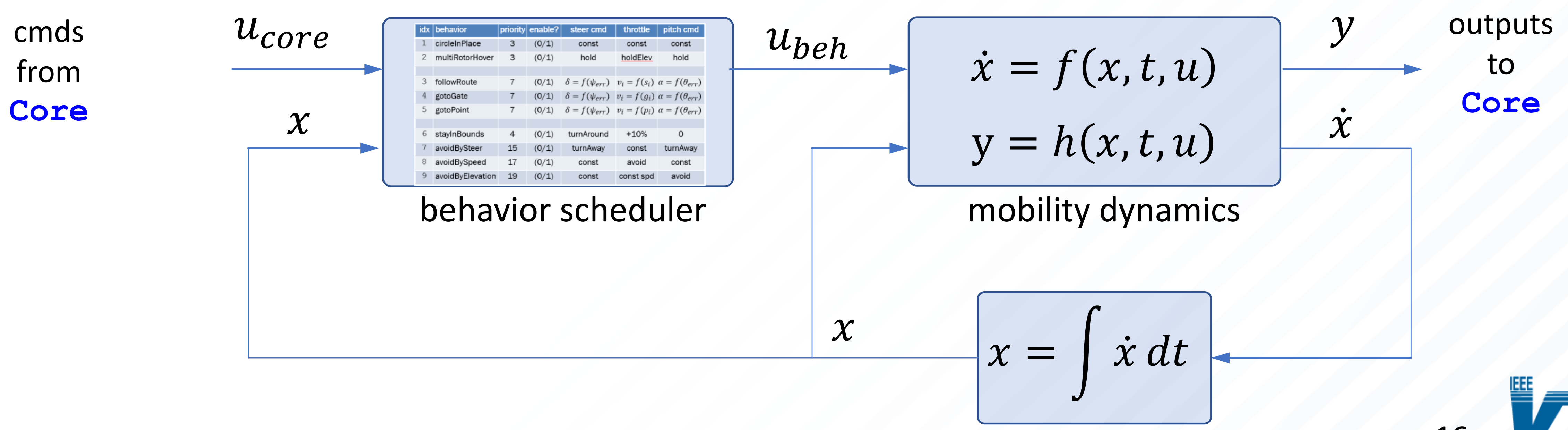
Vehicle inputs, u_{beh}

idx	behavior	priority	enable?	steer cmd	throttle	pitch cmd
1	circleInPlace	3	(0/1)	const	const	const
2	multiRotorHover	3	(0/1)	hold	holdElev	hold
3	followRoute	5	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(s_i)$	$\alpha = f(\theta_{err})$
4	gotoGate	6	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(g_i)$	$\alpha = f(\theta_{err})$
5	gotoPoint	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(p_i)$	$\alpha = f(\theta_{err})$
6	stayInBounds	20	(0/1)	turnAround	+10%	0
7	avoidBySteer	15	(0/1)	turnAway	const	turnAway
8	avoidBySpeed	17	(0/1)	const	avoid	const
9	avoidByElevation	19	(0/1)	const	const spd	avoid

[1] Brooks, R. A. (1990). Elephants don't play chess. *Robotics and autonomous systems*, *6*(1-2), 3-15., https://www2.cs.sfu.ca/~vaughan/teaching/894/papers/elephants.pdf

MoVE Built-in Vehicle Model

Vehicle models with behaviors, mobility dynamics, and RK4 integrator are advanced in soft-real-time

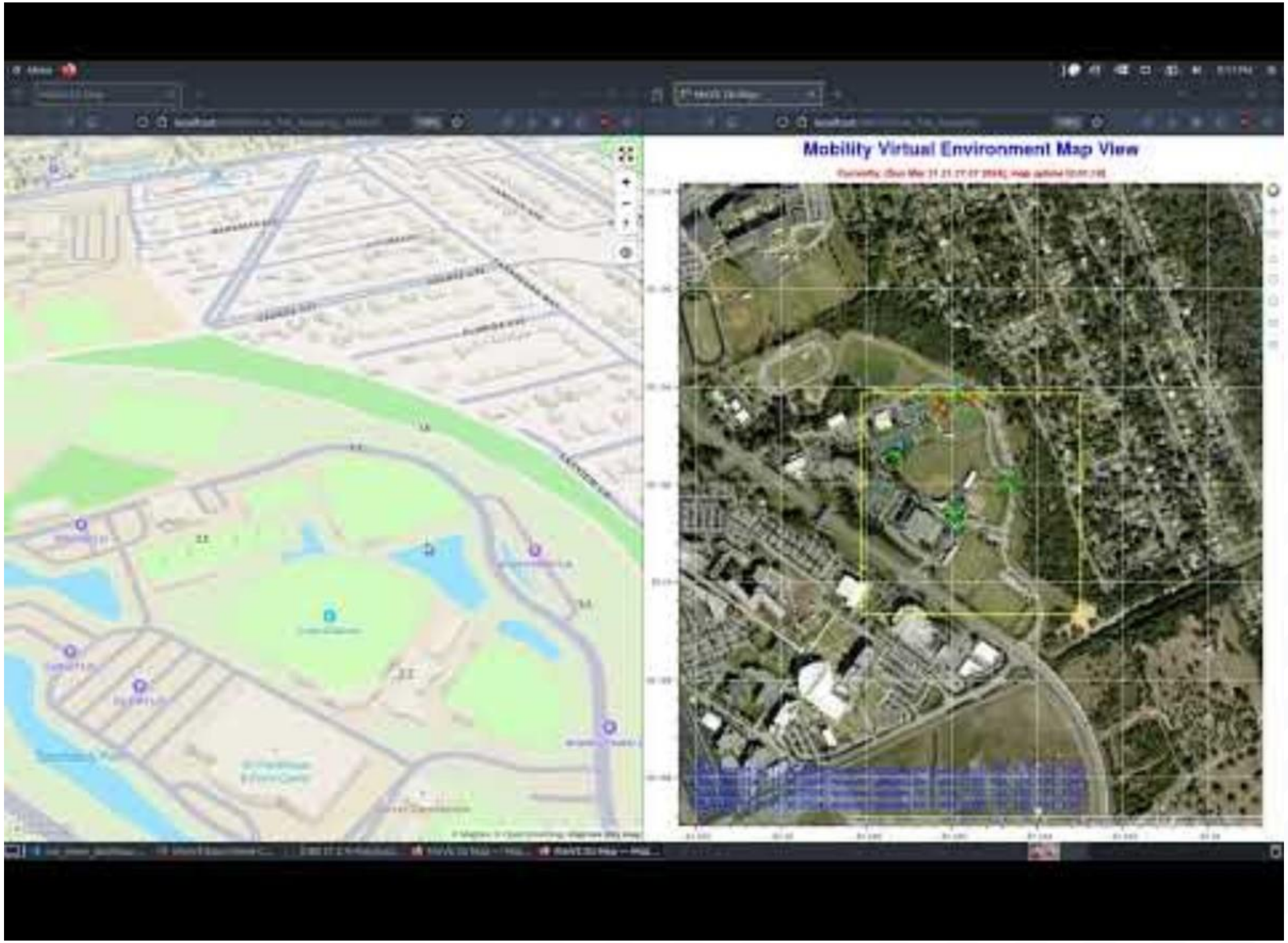


idx	behavior	priority	enable?	steer cmd	throttle	pitch cmd
1	circleInPlace	3	(0/1)	const	const	const
2	multiRotorHover	3	(0/1)	hold	holdElev	hold
3	followRoute	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(s_i)$	$\alpha = f(\theta_{err})$
4	gotoGate	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(g_i)$	$\alpha = f(\theta_{err})$
5	gotoPoint	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(p_i)$	$\alpha = f(\theta_{err})$
6	stayInBounds	4	(0/1)	turnAround	+10%	0
7	avoidBySteer	15	(0/1)	turnAway	const	turnAway
8	avoidBySpeed	17	(0/1)	const	avoid	const
9	avoidByElevation	19	(0/1)	const	const spd	avoid

MoVE Video of: default.cfg

Behaviors enabled in **default.cfg**

```
{'wander'        : 1,
 'periodicTurn': 2,
 'stayInBounds': 10,
 'avoidBySteer': 15 }
```



https://youtu.be/tYYYZgfoXZE

v2v

Vehicle-To-Vehicle Communication

- A very simple broadcast model of v2v communications is implemented.
- udp/ip multicast is a simple way to model a broadcast network among all vehicles
- New threads:
  - **`v2vUpdateToNetwork()`**
  - **`v2vUpdateFromNetwork()`**
- ¡Voila!   v2v communications!

v2v table

vid	name	lat	lon	elev	M.S.	% complete
100	joe			0	1	0.1
101	jim			10	5	0.8
102	bill			20	7	0.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Each vehicle reports and receives v2v messages and builds it's own table of what every other vehicle (and pedestrian) is doing

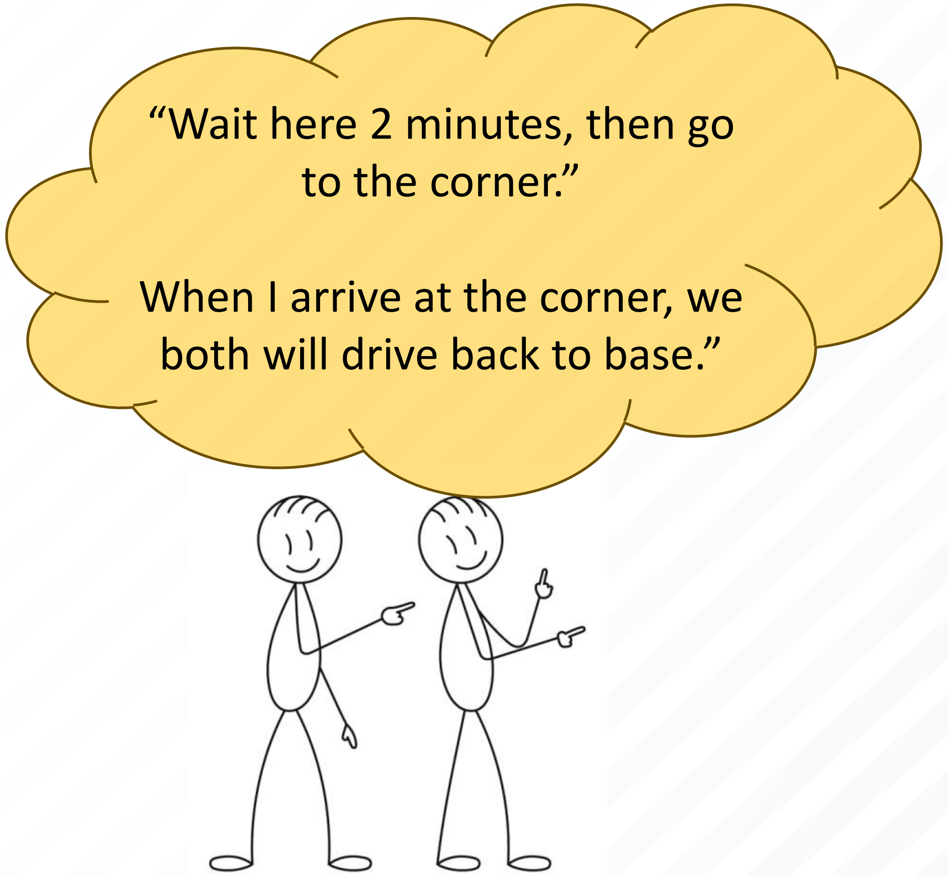
a Mission Sequencer

A Multi-Vehicle Mission Sequencer

- Multi-vehicle **missions** are more than behaviors and communications.

- Missions include time and space-dependent conditions across multiple vehicles, with goals for all vehicles.

- A mission for a single vehicle needs to be described as a sequence of steps that can be completed – to complete the mission.

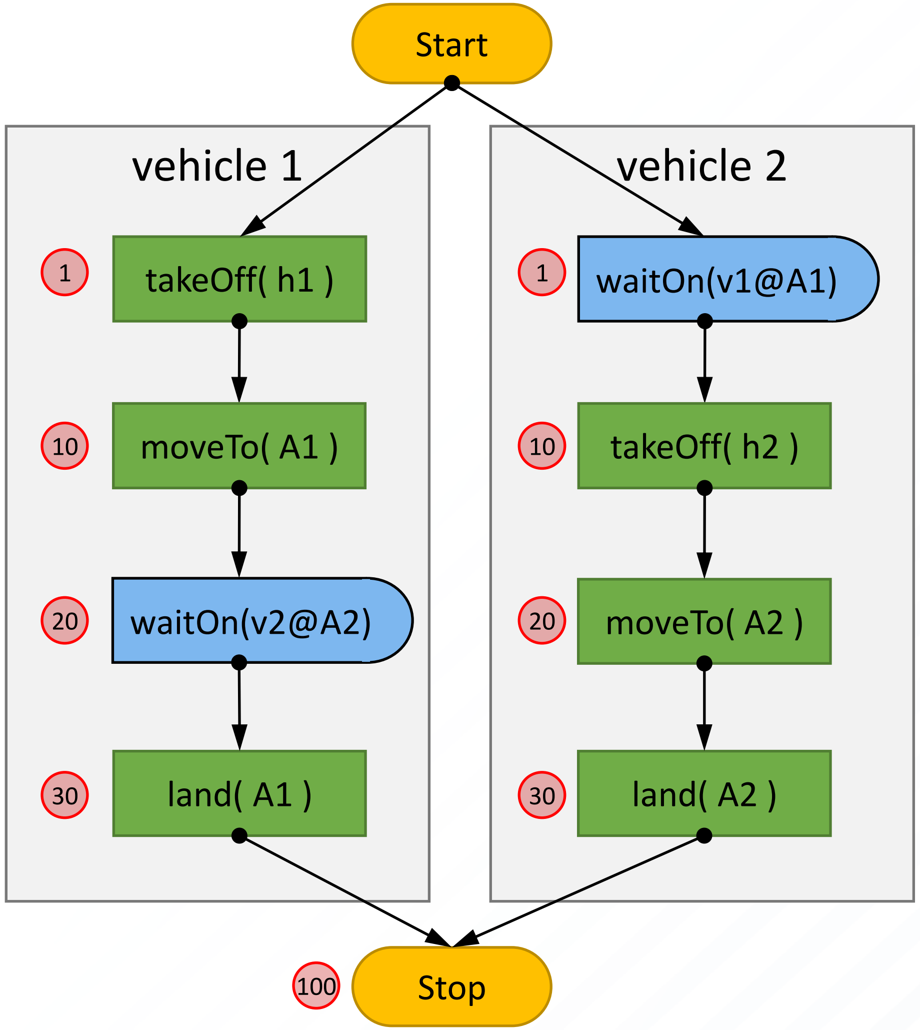


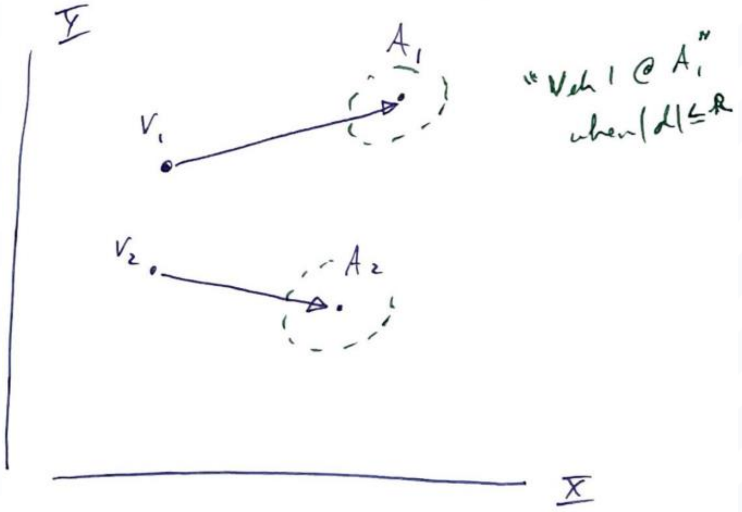
The Multi-Vehicle Mission Language (MML)

Mission Commands for Both:

- vid100:
  - takeOff()
  - moveTo(A1)
  - waitUntil(vid101 @ A2)
  - land()

- vid101:
  - waitUntil(vid100 @ A1)
  - takeOff()
  - moveTo(A2)
  - land()





a Mission example:

missionDefault.cfg

missionDefault.cfg

```
# a mission table is constructed from rows of commands, for a particular vehicle identifier (vid)
# labels prior to the ':' have no influence but must be unique; they do not need to be ordered
# each vid's set of commands are read (in any order) and sorted by missionState
# required: 'vid', 'missionState', 'action'; all others may or may not contain action-specific keys
[missionCommands]
# Newton
missionCmd00:  {'vid':100, 'missionState':   2 , 'action':'waitElapsed',       'eTimeThresh'    :  4.0 } # (s) wait this many elapsed seconds within the mission sequencer

missionCmd01:  {'vid':100, 'missionState': 10 , 'action':'goToPoint',         'point'          : 'first',   'speed': 3, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd02:  {'vid':100, 'missionState': 15 , 'action':'waitElapsed',       'eTimeThresh'    :  4.0 } # (s) wait this many elapsed seconds within the mission sequencer

missionCmd03:  {'vid':100, 'missionState': 20 , 'action':'goToPoint',         'point'          : 'second',   'speed': 3, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd04:  {'vid':100, 'missionState': 25 , 'action':'waitElapsed',       'eTimeThresh'    :  4.0 } # (s) wait this many elapsed seconds within the mission sequencer

missionCmd05:  {'vid':100, 'missionState': 30 , 'action':'goToPoint',         'point'          : 'third',   'speed': 3, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd06:  {'vid':100, 'missionState': 35 , 'action':'waitElapsed',       'eTimeThresh'    :  4.0 } # (s) wait this many elapsed seconds within the mission sequencer

missionCmd07:  {'vid':100, 'missionState': 40 , 'action':'goToPoint',         'point'          : 'home',   'speed': 3, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd08:  {'vid':100, 'missionState':100 , 'action':'goToMissionState',  'newMissionState':    2 , 'maxIterations': 200 } # 'maxIterations' is max number of goTo returns

missionCmd09:  {'vid':100, 'missionState':200 , 'action':'waitElapsed',       'eTimeThresh'    :  4.0 } # (s) wait this many elapsed seconds within the mission sequencer


# MarieCurie
missionCmd101: {'vid':101, 'missionState':   1 , 'action':'waitOnVehProgress', 'otherVeh'       :  100,     'progThresh': 11.0   } # otherVeh's progress must be >= threshold

missionCmd102: {'vid':101, 'missionState': 10 , 'action':'goToPoint',         'point'          : 'third',   'speed': 5, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd103: {'vid':101, 'missionState': 20 , 'action':'waitOnVehProgress', 'otherVeh'       :  100,     'progThresh': 20.0   } # otherVeh's progress must be >= threshold

missionCmd104: {'vid':101, 'missionState': 30 , 'action':'goToPoint',         'point'          : 'home',   'speed': 5, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }

missionCmd105: {'vid':101, 'missionState': 20 , 'action':'waitOnVehProgress', 'otherVeh'       :  100,     'progThresh': 31.0   } # otherVeh's progress must be >= threshold

missionCmd106: {'vid':101, 'missionState': 30 , 'action':'goToPoint',         'point'          : 'first', 'speed': 5, 'riseRate': 3, 'rThresh': 3.0, 'arrivalType':'stop' }
```

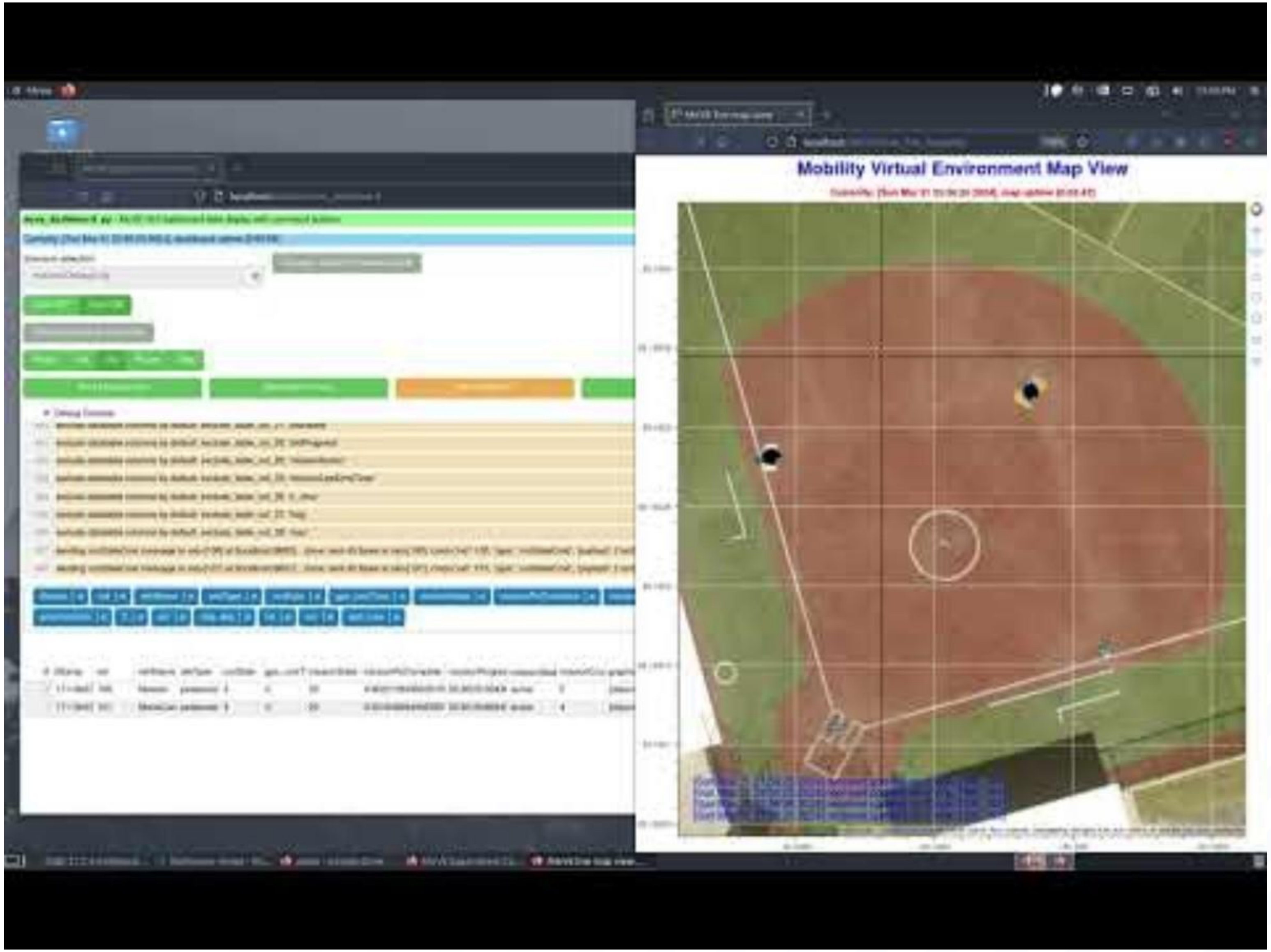
Newton
vid=100

MarieCurie
vid=101

MoVE Video of: default.cfg

Behaviors enabled in
missionDefault.cfg

```
{'multiRotorHover': 1,
 'goToPoint'      : 5,
 'stayInBounds'   : 12,
 'avoidBySteer'   : 10 }
```



https://youtu.be/zNYqOKASJCc

Mission Sequencer Functionality

- What just happened? How did that work?
- Each vehicle's Mission Sequence specifies a ***mission action***, with clear criteria on when it is complete.
- Each mission action has a metric (time or distance) that can be tracked, from 0% to 100% complete.
- When the current mission command is 100% complete, **`missionState`** advances to the next mission command.
- **A complete mission is:** a sequence of ***mission actions***, going from point-to-point, and perhaps waiting a duration, or waiting on another vehicle to arrive somewhere.

Mission Actions:
`goToPoint(pt)`
`waitElapsed(dt)`
`waitOnVehProgress(other,thresh)`
`goToMissionState(newMS)`

The Mission Architecture

missionCmd changes behavior priorities

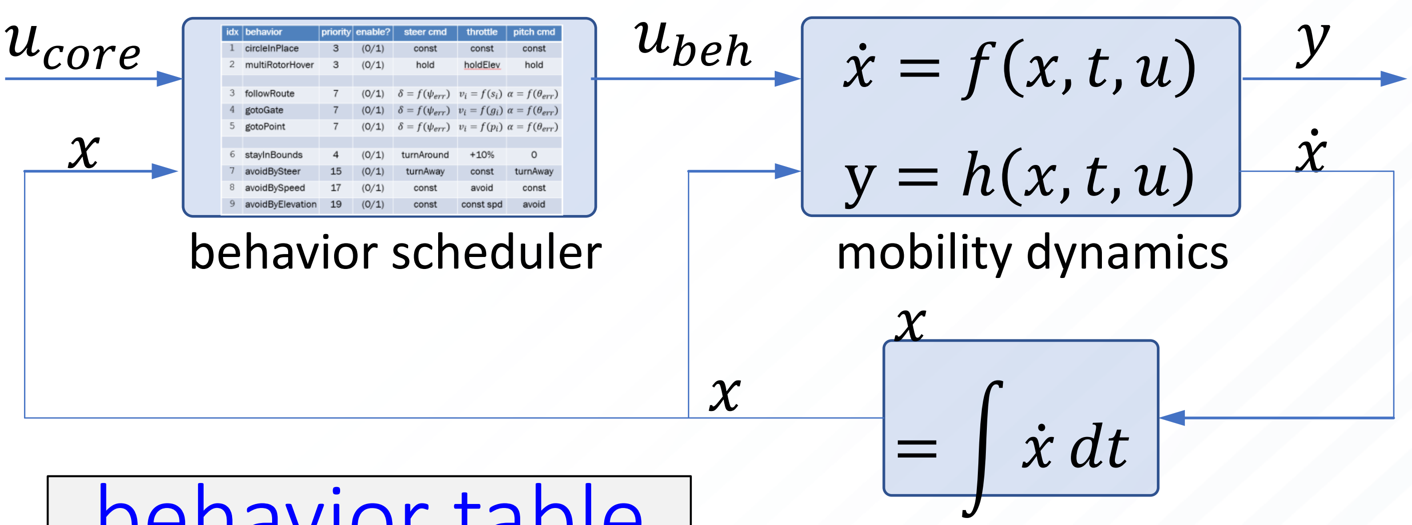
i	[action]	i+1
1	gotoPoint(Pt_1)	2
10	waitElapsed(30s)	11
20	waitOnVehProgress(otherVeh, threshold)	21
30	goToMissionState(100)	31
100	gotoPoint($landPt_1$)	101

v2v table

vid	name	lat	lon	elev	M.S.	% complete
100	joe			0	1	0.1
101	jim			10	5	0.8
102	bill			20	7	0.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮

MoVE

Built-in Vehicle Model



idx	behavior	priority	enable?	steer cmd	throttle	pitch cmd
1	circleInPlace	3	(0/1)	const	const	const
2	multiRotorHover	3	(0/1)	hold	holdElev	hold
3	followRoute	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(s_i)$	$\alpha = f(\theta_{err})$
4	gotoGate	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(g_i)$	$\alpha = f(\theta_{err})$
5	gotoPoint	7	(0/1)	$\delta = f(\psi_{err})$	$v_i = f(p_i)$	$\alpha = f(\theta_{err})$
6	stayInBounds	4	(0/1)	turnAround	+10%	0
7	avoidBySteer	15	(0/1)	turnAway	const	turnAway
8	avoidBySpeed	17	(0/1)	const	avoid	const
9	avoidByElevation	19	(0/1)	const	const spd	avoid

behavior table controls vehicle

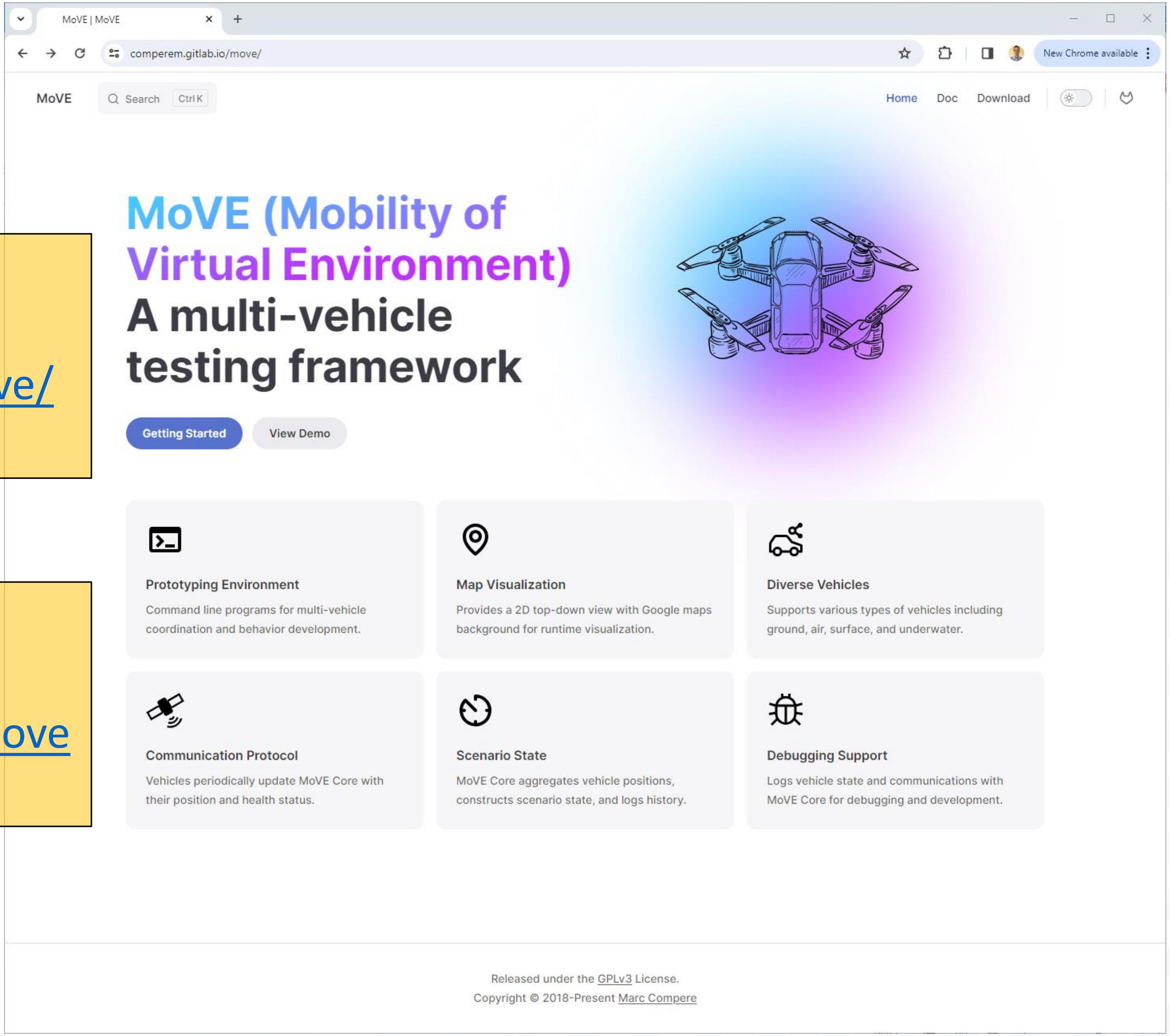
Code
Docs
References

Documentation site:

https://comperem.gitlab.io/move/

Code:

https://gitlab.com/comperem/move



Conclusion

- MoVE started as an open-source project in 2018
- MoVE provides multi-vehicle, multi-domain simulation, with a v2v model, behavior models, and a Multi-Vehicle Mission Sequencer.
- MoVE also accepts ADS-B, Wifi, Xbee, and other networks for collecting live telemetry from multiple (real) vehicles in field tests.
- Questions and inquiries: Marc Compere, comperem@erau.edu !!!

MoVE Papers

Article References

1. Compere, M. D., Adkins, K. A., Krishnan, A. M., Schroeder, R., & James, C. N. (2024). The mobility virtual environment (MoVE): an open source framework for gathering and visualizing atmospheric observations using multiple vehicle-based sensors. *Environmental Science: Atmospheres*., https://pubs.rsc.org/en/content/articlehtml/2024/ea/d2ea00106c
2. Marc Compere, Kevin Adkins, Avinash Muthu-Krishnan, “Go with The Flow: Estimating Wind with Uncrewed Aircraft”, *Drones, Special Issue "Weather Impacts on Uncrewed Aircraft"*, *Drones* **2023**, *7*(9), 564; https://doi.org/10.3390/drones7090564
3. Adkins, K.A.; Becker, W.; Ayyalasomayajula, S.; Lavenstein, S.; Vlachou, K.; Miller, D.; Compere, M.; Muthu Krishnan, A.; Macchiarella, N. Hyper-Local Weather Predictions with the Enhanced General Urban Area Microclimate Predictions Tool. Drones 2023, 7, 428. https://doi.org/10.3390/drones7070428
4. M. Compere, K. Adkins, O. Legon, P. Currier, “MoVE: A Mobility Virtual Environment for Testing Multi-Vehicle Scenarios”, In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019, url: http://gvsets.ndia-mich.org/publication.php?documentID=721
5. M. Compere, G. Holden, O. Legon, “MoVE: A Mobility Virtual Environment for Autonomous Vehicle Testing”, IMECE2019-10936, International Mechanical Engineering Congress and Exposition, Nov. 2019, Salt Lake City UT, link
6. M. Compere, “MoVE: Mobility Virtual Environment released with GPLv3 license as Open-Source, publicly available software”; website url: https://comperem.gitlab.io/move/; source code url: https://gitlab.com/comperem/move, 2018.

IEEE VTS Acknowledgement

Dr. Compere would like to acknowledge the

IEEE Vehicular Technology Society (https://vtsociety.org/)

for sponsoring the

Webinar Series on Advanced Air Mobility!

(https://vtsociety.org/post/announcement/webinar-series-advanced-air-mobility)

Join IEEE VTS at
www.vtsociety.org

Follow IEEE VTS on social media

Website
www.vtsociety.org

Facebook
facebook.com/IEEEVTS

Twitter
@IEEE_VTS

LinkedIn
www.linkedin.com/company/ieee-vehicular-technology-society

Extras

Real Vehicles and Real Network Telemetry

1	Simulated MoVE vehicles
2	GPS updates over cellular network
3	Xbee 2.4Ghz mesh network
4	ADS-B 1090MHz
5	RemoteID, Bluetooth
6	Lora Mesh with CDP
7	Cloud upload