The MOOS is dead. Long live the MOOS!

🎉 Cheers for another 20 years 🎊

Dr. Mohamed Saad Ibn Seddik | msis@blksail.ai | blkSAIL

MOOS-DAWG 22 | August 11th, 2022

Autonomy ≠ Waypoint Following

Autonomy ≠ Unmanned

Change my mind!

Maritime Accidents

CTN/CLS

~3000

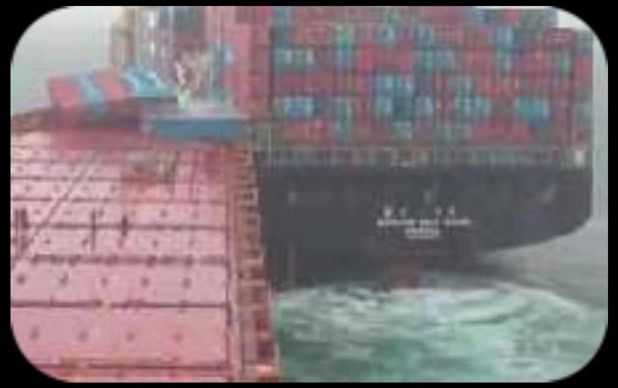
Accidents/Year

USS Fitzgerald/ACX

~$400M/h

93%

Human Error



MSC/Hanjin

Wakashio

1 ship/week

is declared **Total Loss**

Suez Canal

Modular

Data Capture + Decision + Command & Control

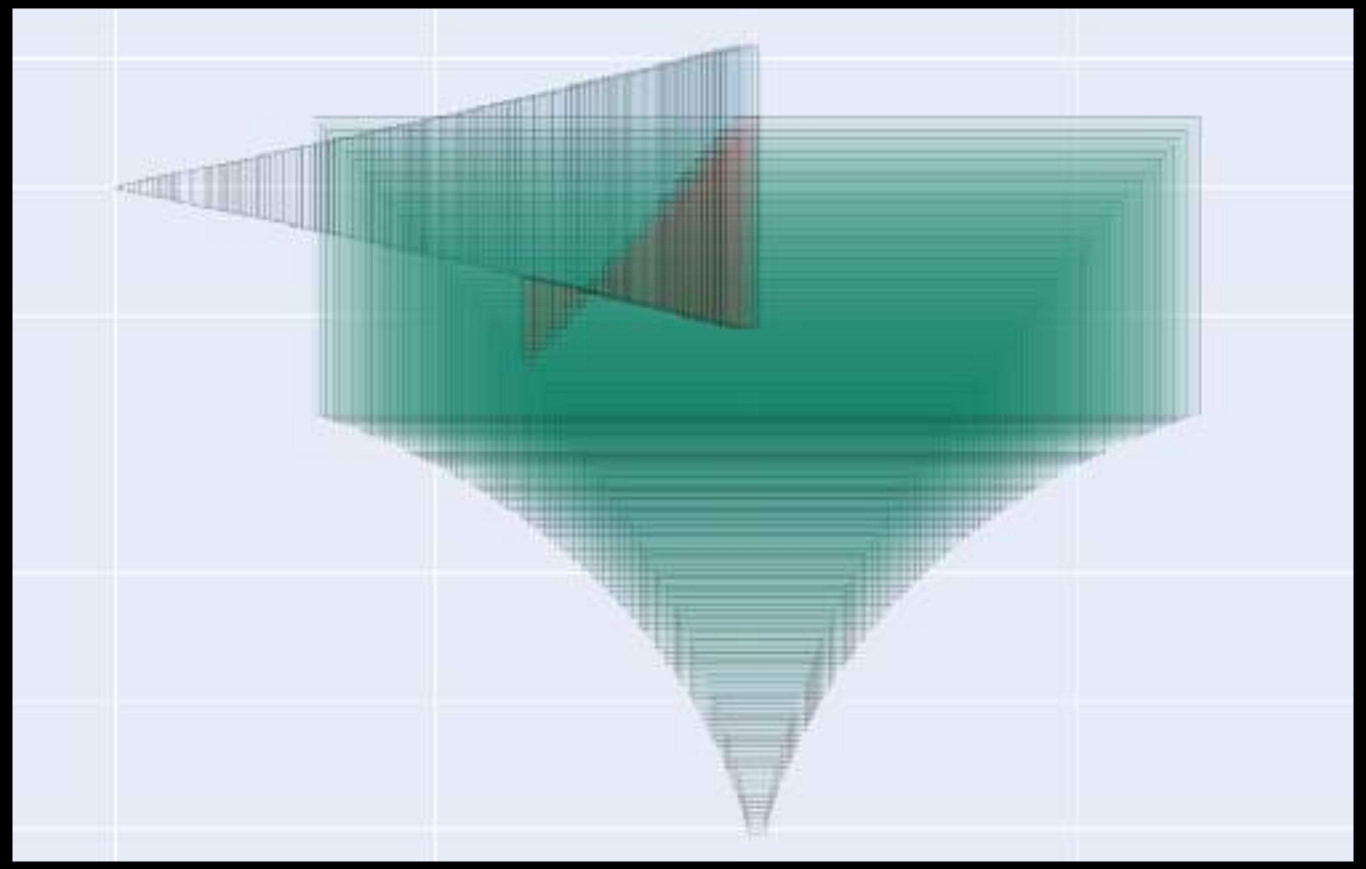
Decision Aid

First Autonomous Ferry w/ COLREGS

Fleet Monitoring Dashboards

BlackSail

Maritime AI as a Service



Guaranteed Collision Detection

Standard Interfaces

$0

CapEx

100%

COTS








SEA MACHINES








“MOOS is easy and straight forward”

David Battle - August 10th, 2022

Challenges

Where MOOS falls (really) short

- Multi language/platform support
  - Web
  - Mobile
  - Micro-controllers
- Cybersecurity
  - Encryption
  - Access control
  - Quality-of-Service

Layer			Protocol data unit (PDU)	Function[23]
Host layers	7	Application	Data	High-level protocols such as for resource sharing or remote file access, e.g. HTTP.
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Transmission of data frames between two nodes connected by a physical layer
	1	Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

OSI Model

Layer			
Host layers	7	Application	← CMOOSAppCasting
	6	Presentation	← CMOOSAsyncCommClient
	5	Session	← XPCTcpSocket
	4	Transport	
Media layers	3	Network	
	2	Data link	
	1	Physical	

CMOOSAppCasting

CMOOSAsyncCommClient

XPCTcpSocket

```
#include "MOOS/libMOOS/Comms/MOOSAsyncCommClient.h"
```

MOOSApp.h

```
#include "MOOS/libMOOS/Comms/MOOSCommClient.h"
```

MOOSAsyncCommClient.h

```
class XPCTcpSocket;
```

MOOSCommClient.h

```
class XPCTcpSocket;
```

CMOOSCommClient.h

Genius!

IT MEANS FREEDOM

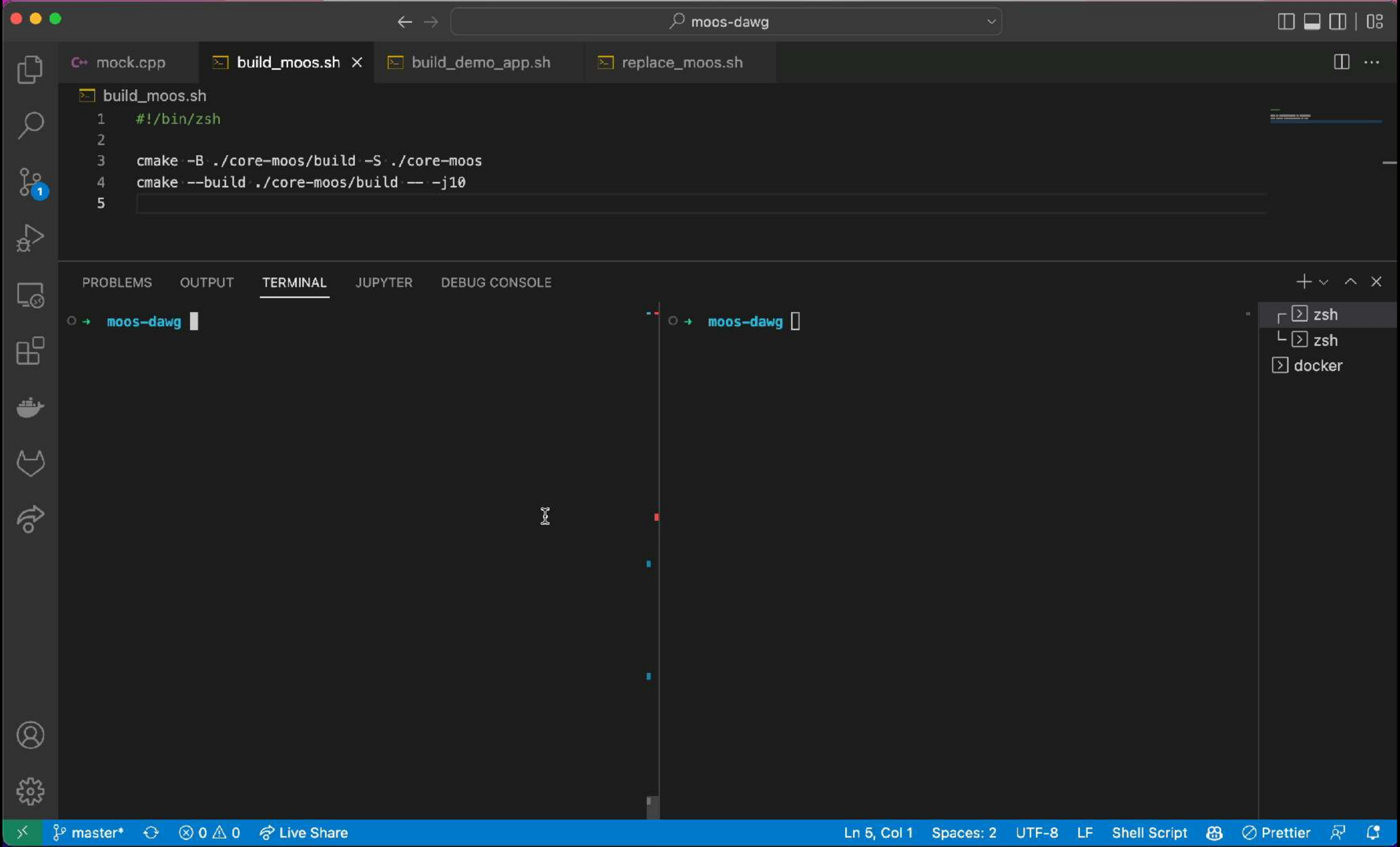
The Plan

1. Write a new industrial Session layer*.
2. Name it: `XPCTcpSocket`. 🤭
3. Rebuild.

*Hint: It is MQTT

DEMO

Or a video if it doesn’t work...


moos-dawg
mock.cpp
build_moos.sh
build_demo_app.sh
replace_moos.sh
build_moos.sh
#!/bin/zsh
cmake -B ./core-moos/build -S ./core-moos
cmake --build ./core-moos/build -- -j10
PROBLEMS
OUTPUT
TERMINAL
JUPYTER
DEBUG CONSOLE
→ moos-dawg
→ moos-dawg
zsh
zsh
docker
master*
0 0
Live Share
Ln 5, Col 1
Spaces: 2
UTF-8
LF
Shell Script
Prettier

2001: MOOS

2012: MOOS v10

2022: **MOOS BS** 💩

https://moos-dawg.blksail.ai/

Is it Open Source?

If Core MOOS is GPL,
then MOOS BS is GPL 🤷‍♂️

Any limitations?

If MOOS is `Notify` and `Register` to you, then no ✅

If you know of `MOOS_TYPE`, then yes.

(Which breaks `pLogger` 😅)

But wait!

There's a `pLoggerBS`, and `pLogSQL`* 💁

*: Supports SQLite and PostgreSQL

LibBS

No GPL Code

100%

MOOS Compatible

Experimental Support

DDS & Zero-Copy IPC (>1ms Data Exchange)

pHorns

`pAntler` with a worst name

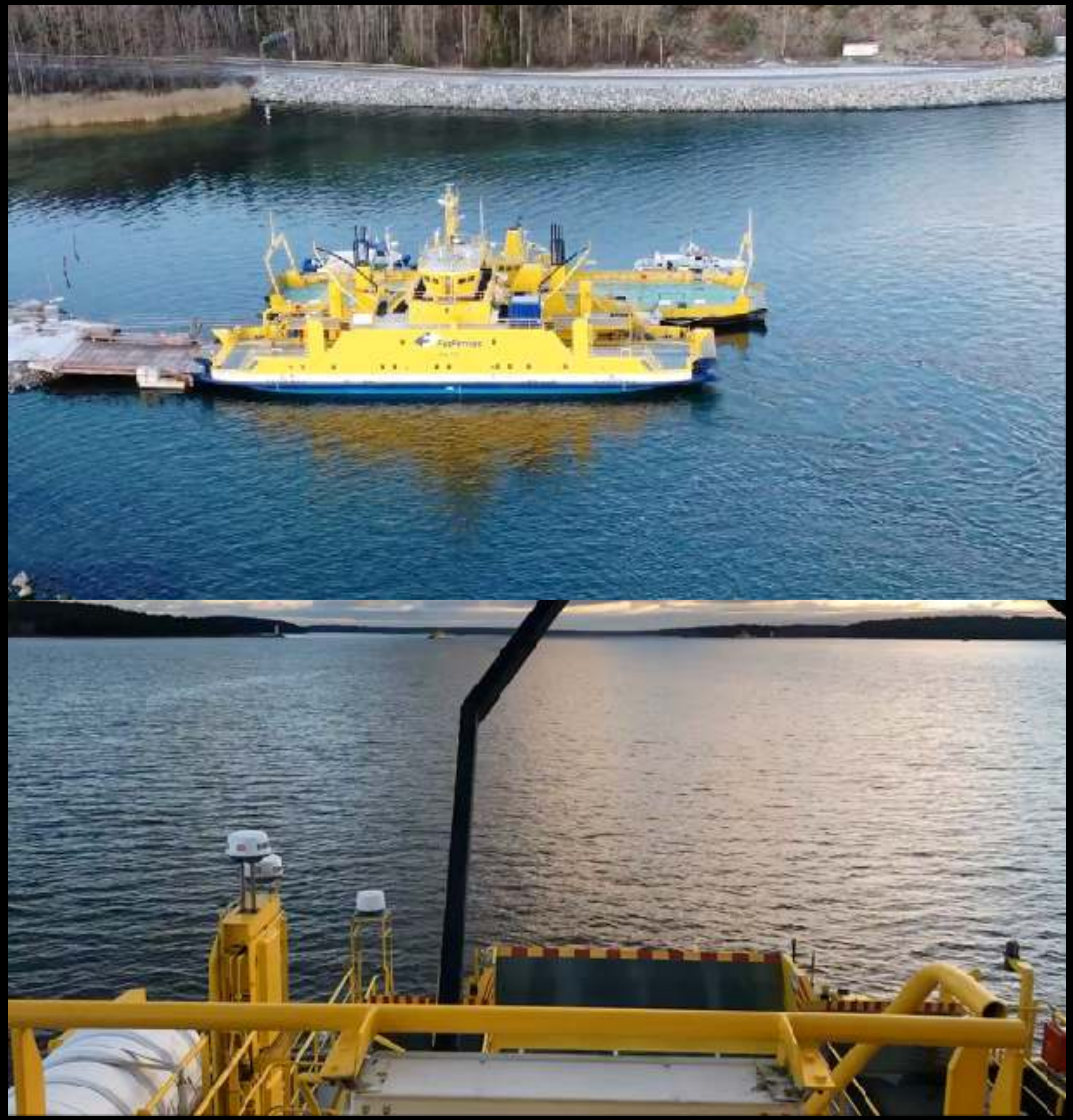
=> MOOS, YAML, TOML config

=> `inotify` relaunches apps if config changes

Modular

Data Capture + Decision + Command & Control

Decision Aid

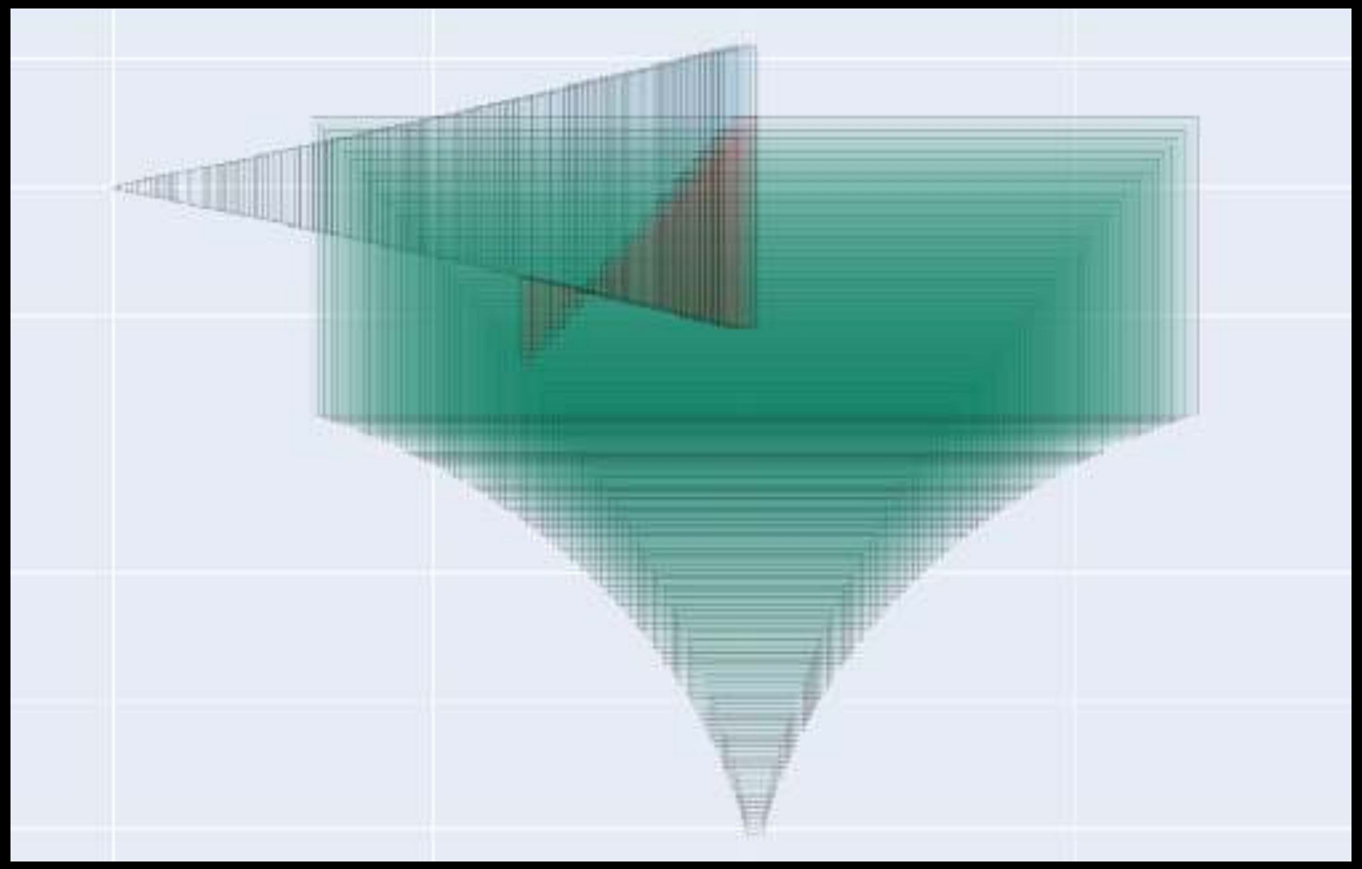


First Autonomous Ferry w/ COLREGS

Fleet Monitoring Dashboards

BlackSail

Maritime AI as a Service



Guaranteed Collision Detection

Standard Interfaces

$0

CapEx

100%

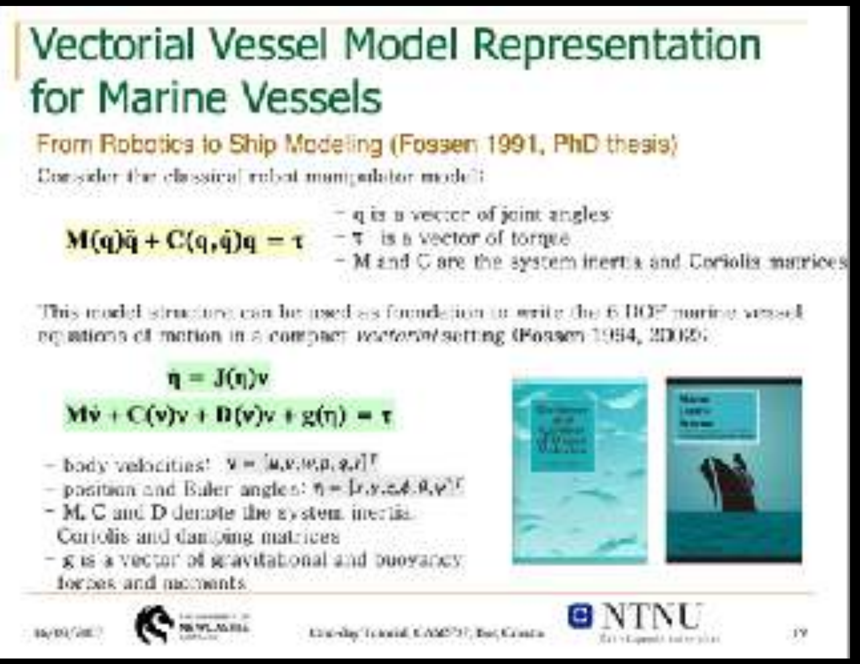
COTS

Thanks!

PSA: No more deer 🦌 ! New “Moose” emoji coming in 2023. You’re welcome!

Dr. Mohamed Saad Ibn Seddik | msis@blksail.ai | blkSAIL

BACKUP SSDs
NVIDIA AGX XAVIER
VDR Interface
NMEA2000 Interface
DO NOT FORMAT
DO NOT FORMAT
DO NOT FORMAT
DO NOT FORMAT
DO NOT FORMAT
SHIP TO: BLKSAIL,1 BROADWAY
14TH FLOOR,CAMBRIDGE MA,02142
NVIDIA
NVIDIA



Computing capture tubes

Luc Jaulin[1], Daniel Lopez[2], Vincent Le Doze[2], Stéphane Le Menec[2], Jordan Ninin[1], Gilles Chabert[3], Mohamed Saad Ibnseddik[1], Alexandru Stancu[4]

[1] Lab STICC, ENSTA Bretagne

[2] MBDA-Airbus Group

[3] Ecole des Mines de Nantes

[4] School of Electrical and Electronic Engineering, Control Systems Research Group, University of Manchester, M139PL, UK

Abstract. Many mobile robots such as wheeled robots, boats, or plane are described by nonholonomic differential equations. As a consequence, they have to satisfy some differential constraints such as having a radius of curvature for their trajectory lower than a known value. For this type of robots, it is difficult to prove some properties such as the avoidance of collisions with some moving obstacles. This is even more difficult when the initial condition is not known exactly or when some uncertainties occur. This paper proposes a method to compute an enclosure (a *tube*) for the trajectory of the robot in situations where a guaranteed interval integration cannot provide any acceptable enclosure. All properties that are satisfied by the tube (such as the non-collision) will also be satisfied by the actual trajectory of the robot.

Keywords: capture tube, contractors, interval arithmetic, robotics, stability.

1 Introduction

A dynamic system can generally be described a state equation of the form:

$$\mathcal{S}_f : \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t). \quad (1)$$

In the situation where the system is uncertain, the state equation becomes a time dependent differential inclusion:

$$\mathcal{S}_F : \dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), t). \quad (2)$$

Validation of the stability properties of such systems is an important and difficult problem [15]. Most of the time, this problem can be transformed into proving the inconsistency of a *constraint network*. For invariant systems (i.e., f or $\mathbf{F}$ do not depend on t), it has been shown [10] that the *V-stability* approach combined with interval analysis [16] can solve the problem efficiently. Here, we extend this work to systems where $\mathbf{f}$ depends on time. Moreover, we will show how to compute a *capture tube*, i.e., a set-valued function which associate to each t a subset of $\mathbb{R}^n$ and such that a feasible trajectory cannot escape. For this, we will need to combine guaranteed integration and Lyapunov theory, such as in [19] or [13], in order to compute this capture tube.

The paper is organized as follows. Section 2 defines the notion of capture tube, which is a specific set of trajectories that encloses the unknown trajectory for