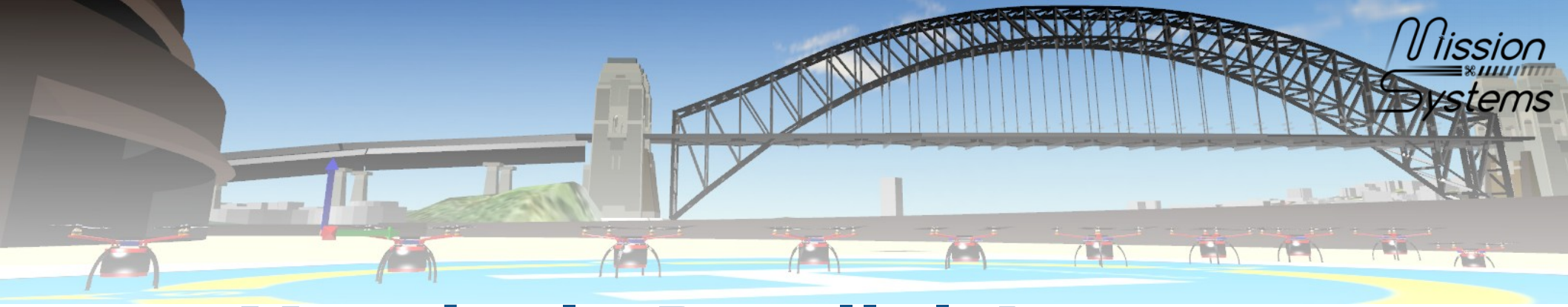


Massively Parallel Autonomy Simulations using MOOS-IvP

David Battle[1], David Johnson[1], Rob Fitch[2]

[1] Mission Systems Pty Ltd, [2] University of Technology Sydney

August 2019

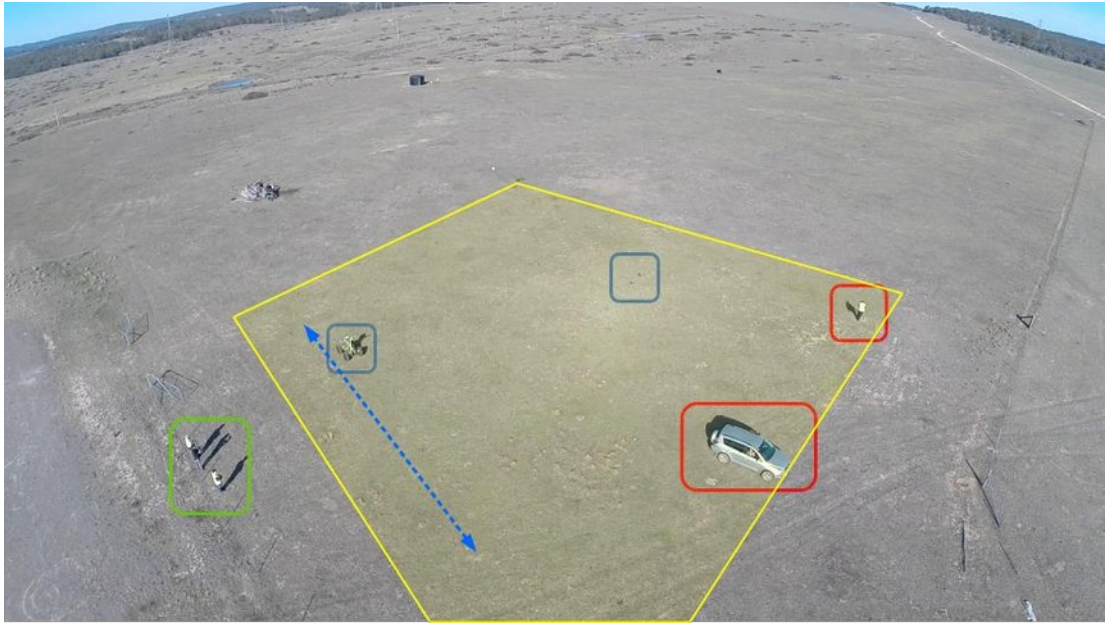
www.missionsystems.com.au

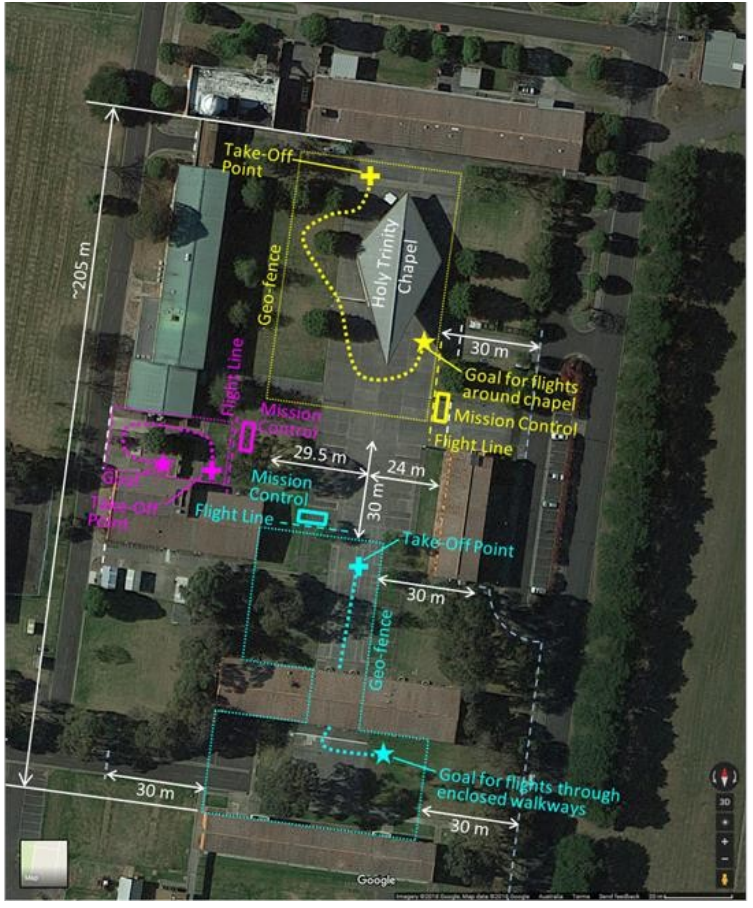
Topics today

- Introduction: Multi-vehicle simulation with MOOS-IvP
- De-centralised ISR project
- MORSE – Modular OpenRobots Simulator Engine
- MOOS – MORSE – PX4 integration
- Sensor simulation with NVIDIA Optix
- Conclusion

De-centralised ISR project

- We want to find a target of interest with a team of UAVs.
- When the target is found, the object is identified in detail.
- The target is in a cluttered environment.
- UAVs must satisfy a safety requirement.





2019 Best Paper Award in Service Robotics: Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch "Multi-Robot Region-of-Interest Reconstruction with Dec-MCTS"

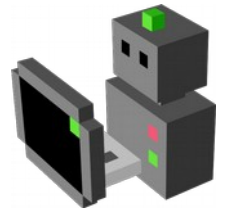
Vehicle for de-centralised ISR

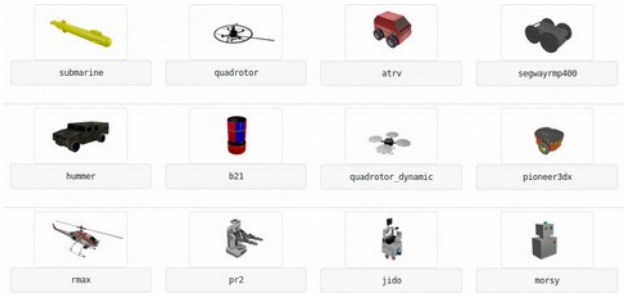
- Standard S500 frame
- 2 x Realsense cameras
- 1 x Ouster OS-1 16-beam lidar
- Nvidia TX2 payload computer
- All up weight 2.5 – 2.8 kg
- Endurance exceeding 10 min

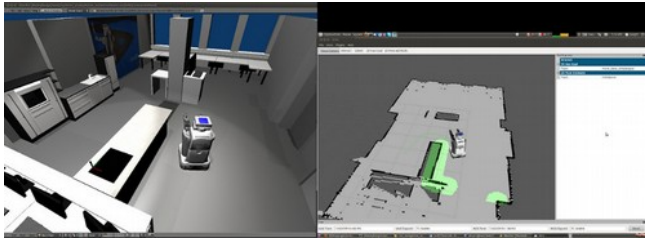
Simulator model: High-poly models appear to have little impact on our simulator performance!

MORSE – Modular OpenRobots Simulator Engine

- Started as a LAAS - CNRS research effort (France) – over 20 contributors now
- Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, Séverin Lemaignan
- MORSE is a generic simulator for academic robotics. It focuses on **realistic 3D simulation** of small to large environments, indoor or outdoor, with one to tens of **autonomous robots**.
- MORSE can be entirely controlled from the command-line. Simulation scenes are generated from simple **Python scripts**.
- MORSE comes with a set of **standard sensors** (cameras, laser scanner, GPS, odometry,...), actuators (speed controllers, high-level waypoints controllers, generic joint controllers) and robotic bases (quadrotors, ATRV, Pioneer3DX, generic 4 wheel vehicle, PR2,...). New ones can easily be added.
- MORSE rendering is based on the **Blender Game Engine**. The OpenGL-based **Game Engine** supports shaders, provides advanced lightning options, supports multi-texturing, and use the state-of-the-art **Bullet library for physics simulation**.







MORSE – Modular OpenRobots Simulator Engine

- Fully programmable via Python scripting
- Access to environmental parameters via simple game engine API, including ray-casting, collision detection, etc
- Sensors and actuators use local data maps to exchange data with middleware data streams

```python
def default_action(self):

    # Toggle FLS beam visibility
    if self.local_data['status'] == 'OFF':
        if self._visible:
            self.FLS_beam.setVisible(False)
            self._visible = False
        return # No data if FLS is OFF
    else:
        if not self._visible:
            self.FLS_beam.setVisible(True)
            self._visible = True

    # Location of FLS transmitter
    source = self.bge_object.worldPosition

    # Direction of FLS beam (local x-axis)
    vect = self.bge_object.worldOrientation.col[0]

    # Target to hit
    target = source + vect

    # Send a ray out infront of the vehicle
    _, point, _ = self.bge_object.rayCast(target, source, self.max_range)
    logger.debug("FLS points to %s and hits %s" % (target, point))

    if point:
        self.local_data['range'] = self.bge_object.getDistanceTo(point)
    else:
        self.local_data['range'] = float('inf')
```

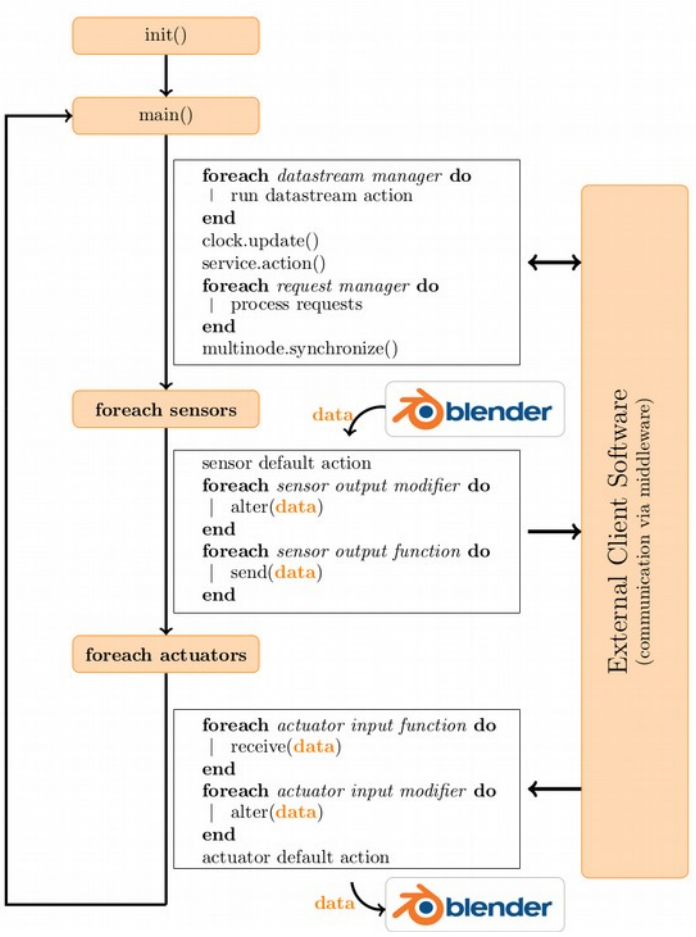


Figure: Degroote et al., 2016

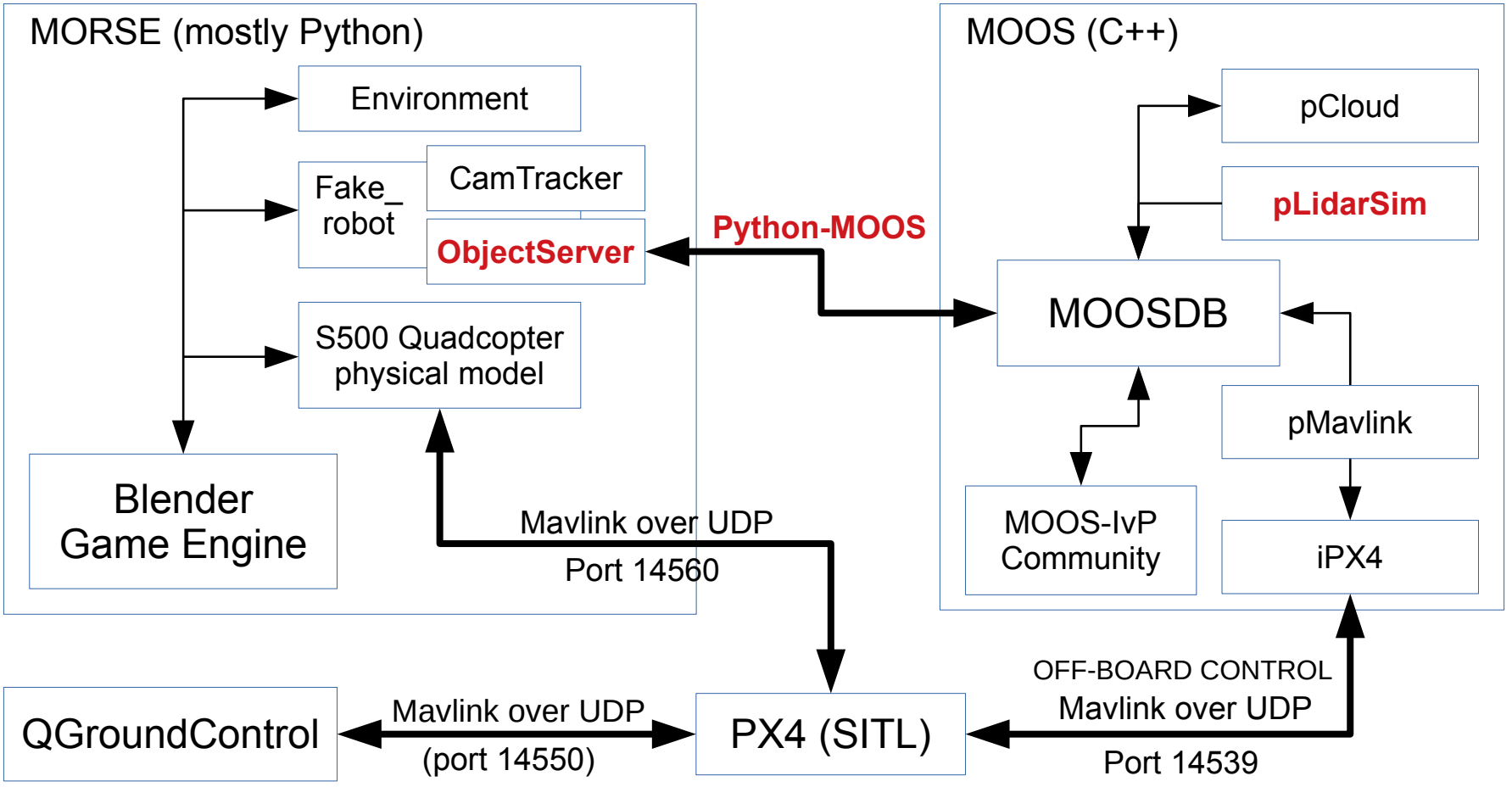
Middleware Support for:

- Sockets
- ROS
- Yarp
- Pocolibs
- MOOS
- HLA
- Text

```python
import logging; logger = logging.getLogger("morse." + __name__)
from morse.middleware.moos import MOOSSubscriber, MOOSNotifier

class FLSReader(MOOSSubscriber):
    """ Read FLS commands and update local data. """

    def initialize(self):

        # initialize the parent class
        MOOSSubscriber.initialize(self)

        # register for control variables from the database
        self.register_message_to_queue('FLS_STATUS','FLS_queue', self.on_FLS_msgs)

    def on_FLS_msgs(self, msg):
        if (msg.key() == 'FLS_STATUS') and (msg.is_string()):
            self.data['status'] = msg.string()

        self._new_messages = True
        return True

class FLSNotifier(MOOSNotifier):
    """ Notify FLS """

    def default(self, ci = 'unused'):
        logger.debug('FLSNotifier is publishing!')
        self.notify('FLS_RANGE', self.data['range'])
```

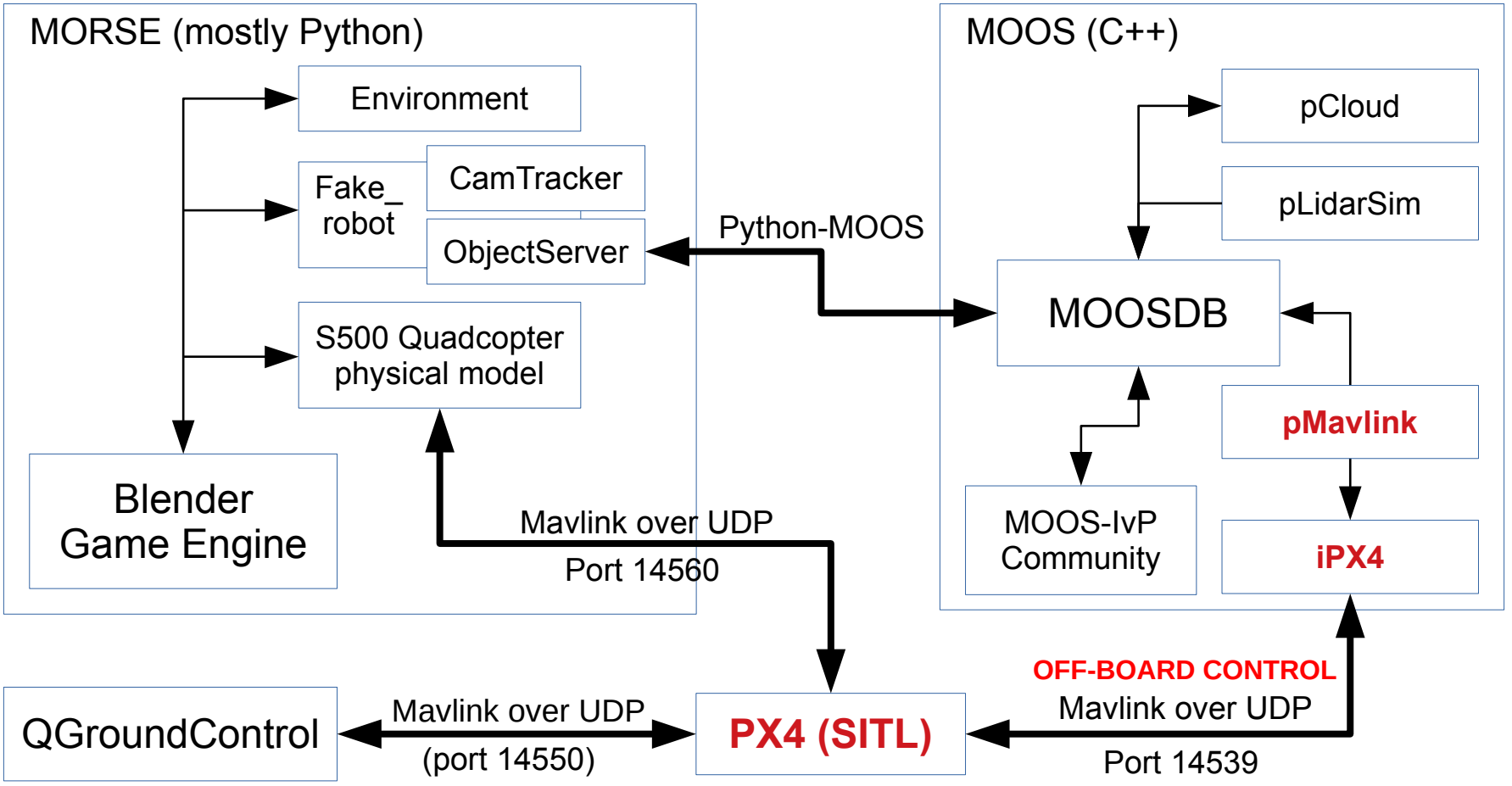
MOOS – MORSE – PX4 integration

With thanks to Saad!

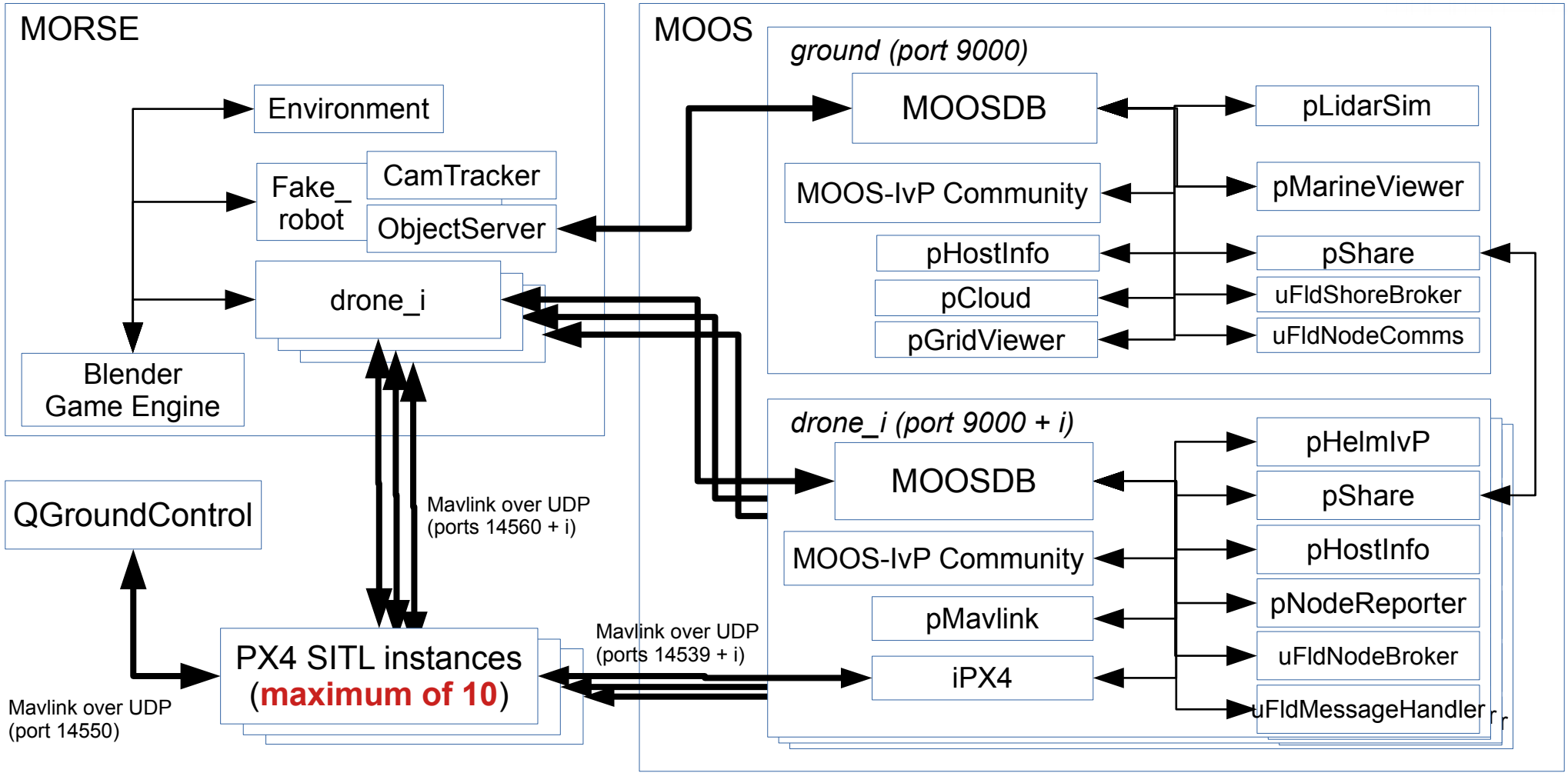


MOOS – MORSE – PX4 integration

With thanks to Muthukumaran Chandrasekaran!



Multi-robot (massively parallel!) simulation



Multi-robot (massively parallel!) simulation

Mission Systems

Single vehicle definition

```
###################################
# Actuators
###################################

self.thrust = Thrust()
self.append(self.thrust)
self.thrust.frequency(200)

###################################
# Sensors
###################################

self.gps = GPS()
self.append(self.gps)
self.gps.level('raw')
self.gps.frequency(10)

self.mag = Magnetometer()
self.append(self.mag)
self.mag.rotate(z = -pi/2)
self.mag.frequency(10)

self.imu = IMU()
self.append(self.imu)
self.imu.frequency(200)

self.asi = Airspeed()
self.append(self.asi)
self.asi.frequency(10)

self.baro = Barometer()
self.append(self.baro)
self.baro.frequency(10)

self.hilsensor = CompoundSensor([self.mag, self.gps, self.imu, self.asi, self.baro])
self.append(self.hilsensor)
self.hilsensor.frequency(200)

self.lidar1 = Lidar()
self.lidar1.translate(z = 0)
self.lidar1.properties(Beam_width_elevation = 16)
self.lidar1.properties(Max_range = 50)
self.lidar1.add_stream('moos','drone_uts_sim.sensors.LidarMoos.lidarReader')
self.lidar1.add_stream('moos','drone_uts_sim.sensors.LidarMoos.lidarNotifier')
self.lidar1.frequency(10)
self.append(self.lidar1)
```

Swarm definition

```
###################################
# Build the vehicles
###################################

for i in range(0, vehicles):
    mavlink_port = 14560 + i
    name = 'quad' + str(i + 1)
    drone = drone_uts(name)
    drone.set_mavlink(port = mavlink_port)
    translate_compound(drone, 581, 1256 + i, 14.2)
    rotate_compound(drone, 0, 0, 0)

# Environmental properties get attached here
fake = FakeRobot()

# tracker = Tracker()
# fake.append(tracker)
# tracker.properties(Standoff = 5)
# tracker.properties(Target = 'quad1')
# tracker.frequency(200)

objectServer = Objectserver()
fake.append(objectServer)
objectServer.add_stream('moos','drone_uts_sim.sensors.ObjectServerMoos.objectServerReader')
objectServer.add_stream('moos','drone_uts_sim.sensors.ObjectServerMoos.objectServerNotifier')
objectServer.frequency(10)

# Sydney CBD (starting at Opera House)
env = Environment('Sydney_CBD_terrain.blend', fastmode = False)
env.set_camera_location([580, 1250, 20])
env.set_camera_rotation([0, 0, 0])
env.properties(latitude = -33.867539, longitude = 151.209206, altitude = 0)
```

Mavlink configuration

```
    ###################################
    # Mavlink configuration
    ###################################

def set_mavlink(self, port = 2):

    self.thrust.add_stream('mavlink', 'drone_uts_sim.actuators.hilThrustMavlink.ThrustReader',
        device = 'udpout:localhost:' + str(port))
    self.hilsensor.add_stream('mavlink', 'drone_uts_sim.sensors.hilSensorMavlink.hilSensor',
        device = 'udpout:localhost:' + str(port))
    self.gps.add_stream('mavlink', 'drone_uts_sim.sensors.gpsMavlink.gps',
        device = 'udpout:localhost:' + str(port))
```

Vehicle DB during flight

New altitude messages

9 Processes 72 Variables

Name	Time	Type	Freq	Source	Community	Value
ALT_AMSL	922.797	D	44.9	pMavlink	drone5	31.00100
ALT_REL	922.797	D	44.9	pMavlink	drone5	16.89800
APPCAST	7.303	$	5.2	pHostInfo	drone5	proc=pHostInfo!@#iter=6!@#node=drone5!@#iter=6!@#messages=!@ PHI HOST IP: 192.168.1.102^[[0m!@
APPCAST_REQ	922.204	$	1.0	pMarineViewer	ground	node=all,app=all,duration=3.0,key=pMarineViewer:groundgen,thresh=run_warning
BAT_REMAINING	922.796	D	4.1	pMavlink	drone5	51.00000
BCM_ALERT_REQUEST	1.003	$	0.0	pHelmIvP	drone5	id=avd, var=CONTACT_INFO, val=name=avd $[VNAME] # contact=$[VNAME], alert_range=50, cpa_range=145
COMMAND_ACK	819.476	D	0.0	pMavlink	drone5	0.00000
DB_CLIENTS	922.207	$	0.5	MOOSDB_drone5	drone5	uMS[Area-51-R4],uFldMessageHandler,uFldNodeBroker,pHostInfo,pNodeReporter,pShare,pHelmIvP,iPX4,pMavlink,
DB_EVENT	840.114	$	0.0	MOOSDB_drone5	drone5	connected=uMS[Area-51-R4]
DB_QOS	922.208	$	0.5	MOOSDB_drone5	drone5	iPX4=0.0770092:0.314951:0.0400543:0.137186,pHelmIvP=0.0779629:3.78895:0.0469685:0.124788,pHostInfo=0
DB_RWSUMMARY	922.208	$	0.5	MOOSDB_drone5	drone5	uMS[Area-51-R4]=&,uFldMessageHandler=APPCAST_REQ:NODE_MESSAGE&APPCAST:UFLDMESSAGEHANDLER_ITER_GA
DB_TIME	922.207	D	0.5	MOOSDB_drone5	drone5	1564808476.55552
DB_UPTIME	922.207	D	0.5	MOOSDB_drone5	drone5	922.20692
DB_VARSUMMARY	922.208	$	0.5	MOOSDB_drone5	drone5	AIS_REPORT 09:59:59.000 (write pending) 0.0 Hz ~
DESIRED_HEADING	1.209	D	0.0	pHelmIvP	drone5	0.00000
IPX4_STATUS	922.137	$	0.5	iPX4	drone5	AppErrorFlag=false,Uptime=921.935,cpuload=2.014,memory_kb=1784,memory_max_kb=1828,MOOSName=iPX4,P
IVPHELM_ALLSTOP	1.209	$	0.0	pHelmIvP	drone5	ManualOverride
IVPHELM_ALLSTOP_DEBUG	922.825	$	4.0	pHelmIvP	drone5	ManualOverride
IVPHELM_CPU	922.825	D	4.0	pHelmIvP	drone5	0.23195
IVPHELM_DOMAIN	1.003	$	0.0	pHelmIvP	drone5	course,0,359,360
IVPHELM_MODESET	1.003	$	0.0	pHelmIvP	drone5	MODE#---,ACTIVE#---,INACTIVE#ACTIVE,STATION-KEEPING#ACTIVE,RETURNING#ACTIVE,PATROLLING#ACTIVE,TRAILIN
IVPHELM_REGISTER	1.003	$	11.8	pHelmIvP	drone5	TRAIL_INFO
IVPHELM_STATE	922.825	$	4.0	pHelmIvP	drone5	PARK
LANDED_STATE	922.796	D	4.1	pMavlink	drone5	2.00000
LOCAL_X_NED	922.797	D	27.0	pMavlink	drone5	0.03541
LOCAL_Y_NED	922.797	D	27.0	pMavlink	drone5	0.18753
LOCAL_Z_NED	922.797	D	27.0	pMavlink	drone5	-16.88894
MAVLINK_RECEIVE	922.885	B	409.8	iPX4	drone5	BINARY DATA [0.036 kB]
MOOS_DEBUG	1.209	$	0.0	pHelmIvP	drone5	pHelmIvP AllStop: ManualOverride
NAV_HEADING	922.797	D	44.9	pMavlink	drone5	40.25000
NAV_LAT	922.797	D	44.9	pMavlink	drone5	-33.85618
NAV_LONG	922.797	D	44.9	pMavlink	drone5	151.21549
NAV_PITCH	922.797	D	90.2	pMavlink	drone5	-0.00993
NAV_ROLL	922.797	D	90.2	pMavlink	drone5	-0.01239
NAV_X	922.797	D	44.9	pMavlink	drone5	580.56084
NAV_Y	922.797	D	44.9	pMavlink	drone5	1263.38006
NAV_YAW	922.797	D	90.2	pMavlink	drone5	0.70256
NAV_Z	922.797	D	44.9	pMavlink	drone5	31.00100
NODE_BROKER_ACK	922.512	$	1.0	uFldShoreBroker	ground	community=qround,hostip=192.168.1.102,port_db=9000,pshare_iroutes=192.168.1.102:9300,timewarp=1,status=
NODE_BROKER_PING_0	922.584	$	0.0	uFldNodeBroker	drone5	community=drone5,hostip=192.168.1.102,port_db=9005,pshare_iroutes=192.168.1.102:9305,timewarp=1,time=1
NODE_REPORT_LOCAL	922.361	$	2.0	pNodeReporter	drone5	NAME=drone5,X=580.59,Y=1263.38,HDG=40.25,LAT=-33.856178,LON=151.215493,TYPE=quadcopter,MODE=PAR

Position relative to takeoff

Usual NAV messages

Mavlink binary data

Ground DB during flight

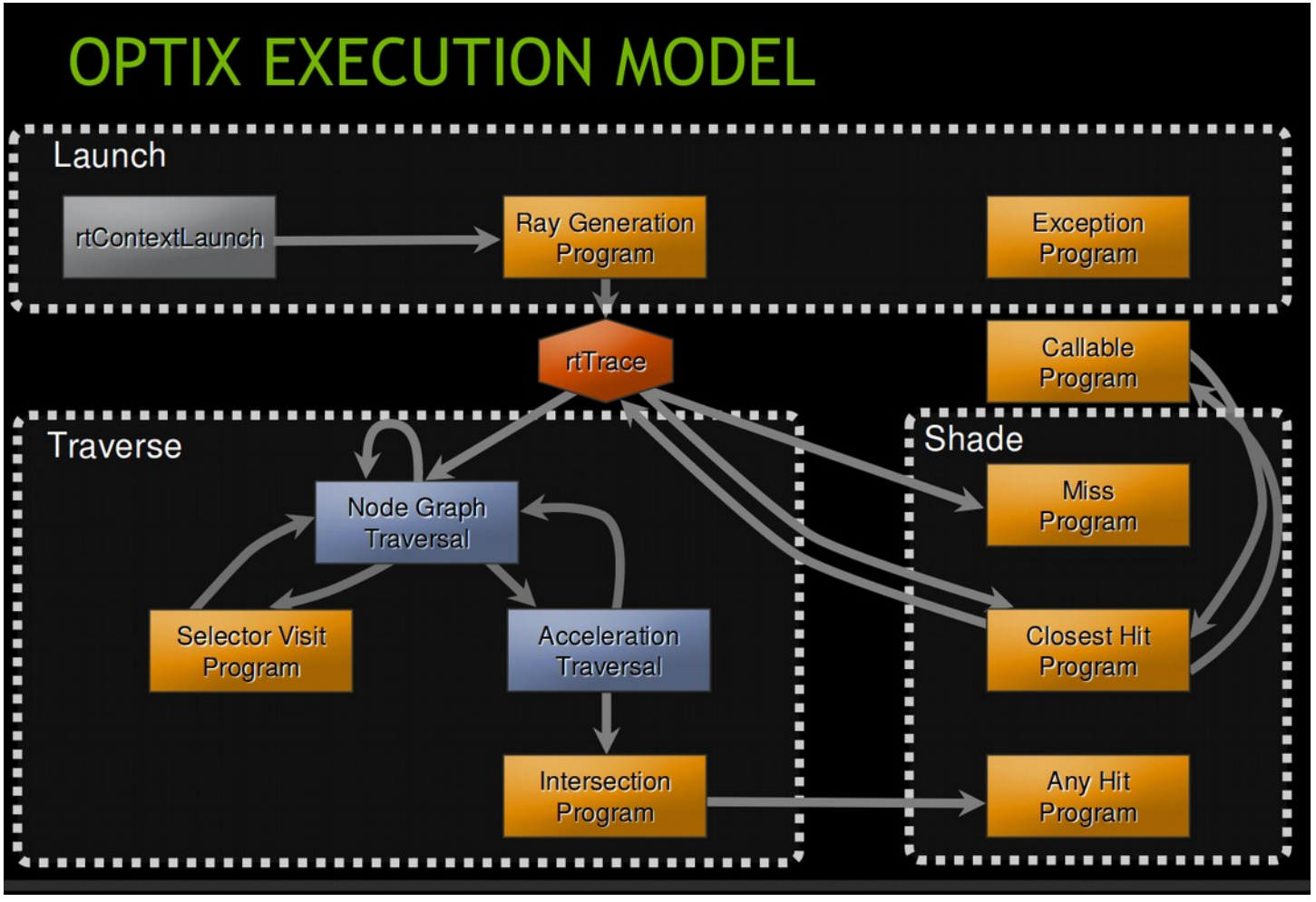
10 Processes 59 Variables

Name	Time	Type	Freq	Source	Community	Value
NODE_BROKER_ACK_DRONE5	69.668	$	1.0	uFldShoreBroker	ground	community=ground,hostip=192.168.1.102,port_db=9000,pshare_iroutes=192.168.1.102:9300,timewarp=1,status=
NODE_BROKER_PING	70.616	$	5.2	uFldNodeBroker	drone4	community=drone4,hostip=192.168.1.102,port_db=9004,pshare_iroutes=192.168.1.102:9304,timewarp=1,time=1
NODE_BROKER_VACK	69.668	$	5.0	uFldShoreBroker	ground	drone5
NODE_REPORT	70.641	$	9.4	pNodeReporter	drone3	NAME=drone3,X=580.46,Y=1261.29,HDG=92.18,LAT=-33.8561968,LON=151.2154916,TYPE=quadcopter,MODE=P
PCLOUD_STATUS	70.467	$	0.5	pCloud	ground	AppErrorFlag=false,Uptime=68.4401,cpuload=58.73,memory_kb=1.186e+05,memory_max_kb=2.748e+05,MOOSN
PGRIDVIEWER_STATUS	70.363	$	0.5	pGridViewer	ground	AppErrorFlag=false,Uptime=67.9344,cpuload=67.82,memory_kb=1.293e+05,memory_max_kb=1.624e+05,MOOSN
PHI_HOST_INFO	61.388	$	0.0	pHostInfo	ground	community=ground,hostip=192.168.1.102,port_db=9000,pshare_iroutes=192.168.1.102:9300,hostip_alts=,timewa
PHI_HOST_IP	61.347	$	0.1	pHostInfo	ground	192.168.1.102
PHI_HOST_IP_ALL	61.388	$	0.0	pHostInfo	ground	192.168.1.102
PHI_HOST_IP_VERBOSE	61.388	$	0.0	pHostInfo	ground	LINUX ANY=192.168.1.102
PHI_HOST_PORT_DB	61.388	$	0.0	pHostInfo	ground	9000
PHOSTINFO_ITER_GAP	70.349	D	1.0	pHostInfo	ground	1.00017
PHOSTINFO_ITER_LEN	70.349	D	1.0	pHostInfo	ground	0.00000
PHOSTINFO_STATUS	70.349	$	0.5	pHostInfo	ground	AppErrorFlag=false,Uptime=69.6627,cpuload=0.04379,memory_kb=4184,memory_max_kb=4184,MOOSName=pHo
PLIDARSIM_STATUS	70.091	$	0.5	pLidarSim	ground	AppErrorFlag=false,Uptime=68.4685,cpuload=9.736,memory_kb=1.223e+06,memory_max_kb=1.567e+06,MOOSN
PMARINEVIEWER_ITER_GAP	70.513	D	4.0	pMarineViewer	ground	1.00084
PMARINEVIEWER_ITER_LEN	70.513	D	4.0	pMarineViewer	ground	0.00006
PMARINEVIEWER_STATUS	69.262	$	0.5	pMarineViewer	ground	AppErrorFlag=false,Uptime=68.9556,cpuload=0.6473,memory_kb=3.558e+04,memory_max_kb=3.558e+04,MOOS
PMV_CONNECT	1.210	D	0.0	pMarineViewer	ground	0.00000
PSHARE_CMD	2.645	$	0.0	uFldShoreBroker	ground	cmd=output,src_name=NODE_MESSAGE_DRONE5,dest_name=NODE_MESSAGE,route=192.168.1.102:9305
PSHARE_INPUT_SUMMARY	70.480	$	1.0	pShare	ground	localhost:9300
PSHARE_OUTPUT_SUMMARY	70.480	$	1.0	pShare	ground	APPCAST_REQ_ALL->APPCAST_REQ:192.168.1.102:9302 & APPCAST_REQ:192.168.1.102:9303 & APPCAST_REQ:192.1
PSHARE_STATUS	69.264	$	0.5	pShare	ground	AppErrorFlag=false,Uptime=68.8298,cpuload=0.3083,memory_kb=4288,memory_max_kb=4288,MOOSName=pSha
QUAD1_LIDAR1_DATA	70.562	B	10.0	pLidarSim	ground	BINARY DATA [135.04 kB]
QUAD1_LIDAR1_TRIGGER	70.560	$	10.0	iMorse	ground	{"lidar_name": "QUAD1_LIDAR1", "max_range": 50, "azim_width": 360.0, "elev_width": 60, "pos": [581.0018920898
QUAD2_LIDAR1_DATA	70.602	B	9.8	pLidarSim	ground	BINARY DATA [132.96 kB]
QUAD2_LIDAR1_TRIGGER	70.600	$	10.1	iMorse	ground	{"lidar_name": "QUAD2_LIDAR1", "max_range": 50, "azim_width": 360.0, "elev_width": 60, "pos": [581.0063476562
QUAD3_LIDAR1_DATA	70.603	B	9.8	pLidarSim	ground	BINARY DATA [132.32 kB]
QUAD3_LIDAR1_TRIGGER	70.600	$	9.8	iMorse	ground	{"lidar_name": "QUAD3_LIDAR1", "max_range": 50, "azim_width": 360.0, "elev_width": 60, "pos": [581.0006103515
QUAD4_LIDAR1_DATA	70.612	B	10.1	pLidarSim	ground	BINARY DATA [129.024 kB]
QUAD4_LIDAR1_TRIGGER	70.600	$	9.8	iMorse	ground	{"lidar_name": "QUAD4_LIDAR1", "max_range": 50, "azim_width": 360.0, "elev_width": 60, "pos": [580.9533691406
QUAD5_LIDAR1_DATA	70.614	B	10.1	pLidarSim	ground	BINARY DATA [128.496 kB]
QUAD5_LIDAR1_TRIGGER	70.600	$	9.8	iMorse	ground	{"lidar_name": "QUAD5_LIDAR1", "max_range": 50, "azim_width": 360.0, "elev_width": 60, "pos": [580.9252929687
SCENE_DATA	70.560	$	0.0	iMorse	ground	{"Inventory": {"Terrain": [[0.0, 0.0, -37.0], [0.0, 0.0, 0.0]], "SkyBox": [[-0.12496918439865112, -12.339553833007
SCENE_QUERY	70.612	$	49.5	pLidarSim	ground	get
UFLDNODECOMMS_ITER_GAP	70.402	D	2.0	uFldNodeComms	ground	1.00096
UFLDNODECOMMS_ITER_LEN	70.440	D	2.0	uFldNodeComms	ground	0.00009
UFLDNODECOMMS_STATUS	70.402	$	0.5	uFldNodeComms	ground	AppErrorFlag=false,Uptime=69.2104,cpuload=0.18,memory_kb=4312,memory_max_kb=4392,MOOSName=uFldNod
UFLDSHOREBROKER_ITER_GAP	70.626	D	1.0	uFldShoreBroker	ground	1.00043
UFLDSHOREBROKER_ITER_LEN	69.669	D	1.0	uFldShoreBroker	ground	0.00011
UFLDSHOREBROKER_STATUS	70.626	$	0.5	uFldShoreBroker	ground	AppErrorFlag=false,Uptime=69.6809,cpuload=0.06738,memory_kb=4212,memory_max_kb=4212,MOOSName=uFld

Lidar messages

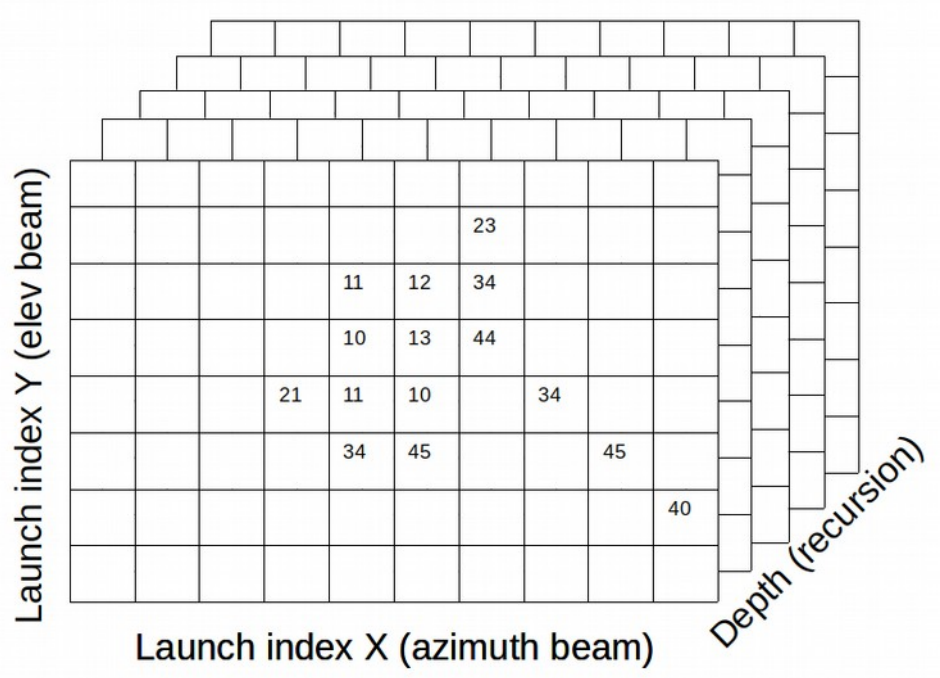
Sensor simulation with NVIDIA Optix

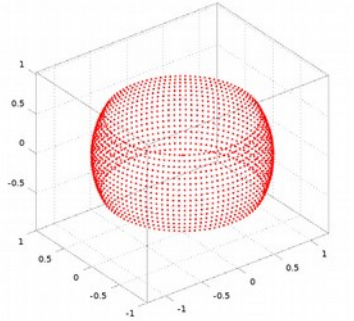
Sensor simulation with NVIDIA Optix



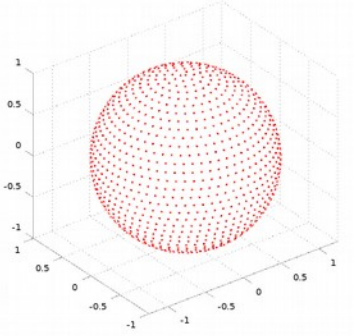
Sensor simulation with NVIDIA Optix

- For lidar simulation, we just return the ray hit coordinates (and maybe a colour!)
- For an acoustic (scalar) signal: 2D or 3D buffer stores complex pressure amplitudes
- For an electromagnetic (vector) signal: 2D or 3D buffer stores complex Jones vectors
- Each complex value represents a generalised delta function at integral sample periods
- Each delta function is “de-jittered” wrt the sample clock via narrow-band phase shifting





Lidar source: Azimuth + Elevation – 2D buffer



Acoustic source: Spherical – 1D buffer

```
struct PerRayData_radiance
{
  float4  point; //   16 bytes
};
```

Lidar payload

```
struct PerRayData_radiance          //  96 bytes
{
  // Local coordinate frame
  frame frame;                      //  36 bytes

  // Curvature matrix
  optix::Matrix2x2 Qre, Qim;        //  32 bytes

  // Complex scalar pressure
  float Pre, Pim;                   //   8 bytes

  // Medium properties
  float kw_re, kw_im;               //   8 bytes
  float rho;                        //   4 bytes

  // Distance of flight
  float dof;                        //   4 bytes

  // Reflection depth
  uint depth;                       //   4 bytes
};
```

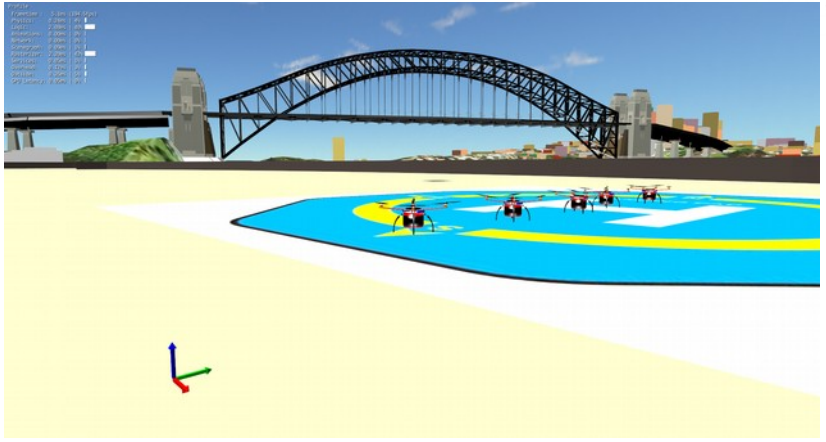
Acoustic payload

```
struct PerRayData_radiance  //  96 bytes
{
  // Local coordinate frame
  frame frame;              //  36 bytes

  // Curvature matrix
  optix::Matrix2x2 Qre,Qim; //  32 bytes

  // Jones vector
  float2 Jre,Jim;           //  16 bytes

  float  dof;               //   4 bytes
  int    depth;             //   4 bytes
  bool   diffract;          //   1 bytes

  short  pad1;              //   2 bytes
  bool   pad2;              //   1 byte
};

#endif /* !__EM_PAYLOAD_H__ */
```

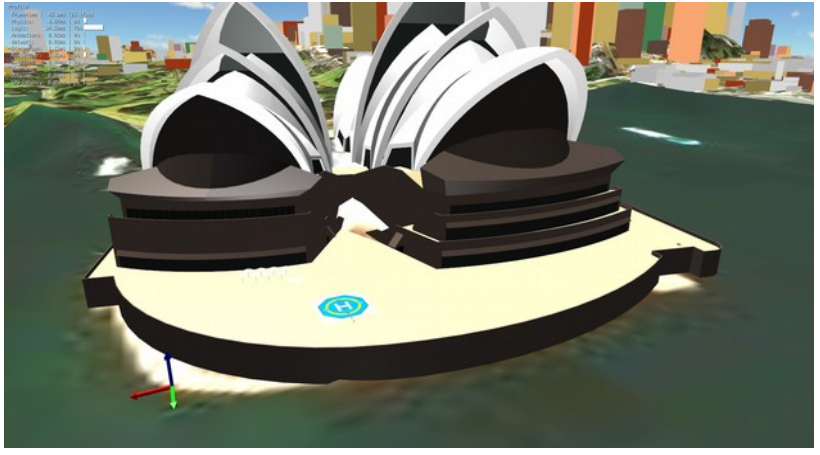
EM (Radar) payload

Ops area for drone simulations

Sydney CBD region – about 6km x 6km



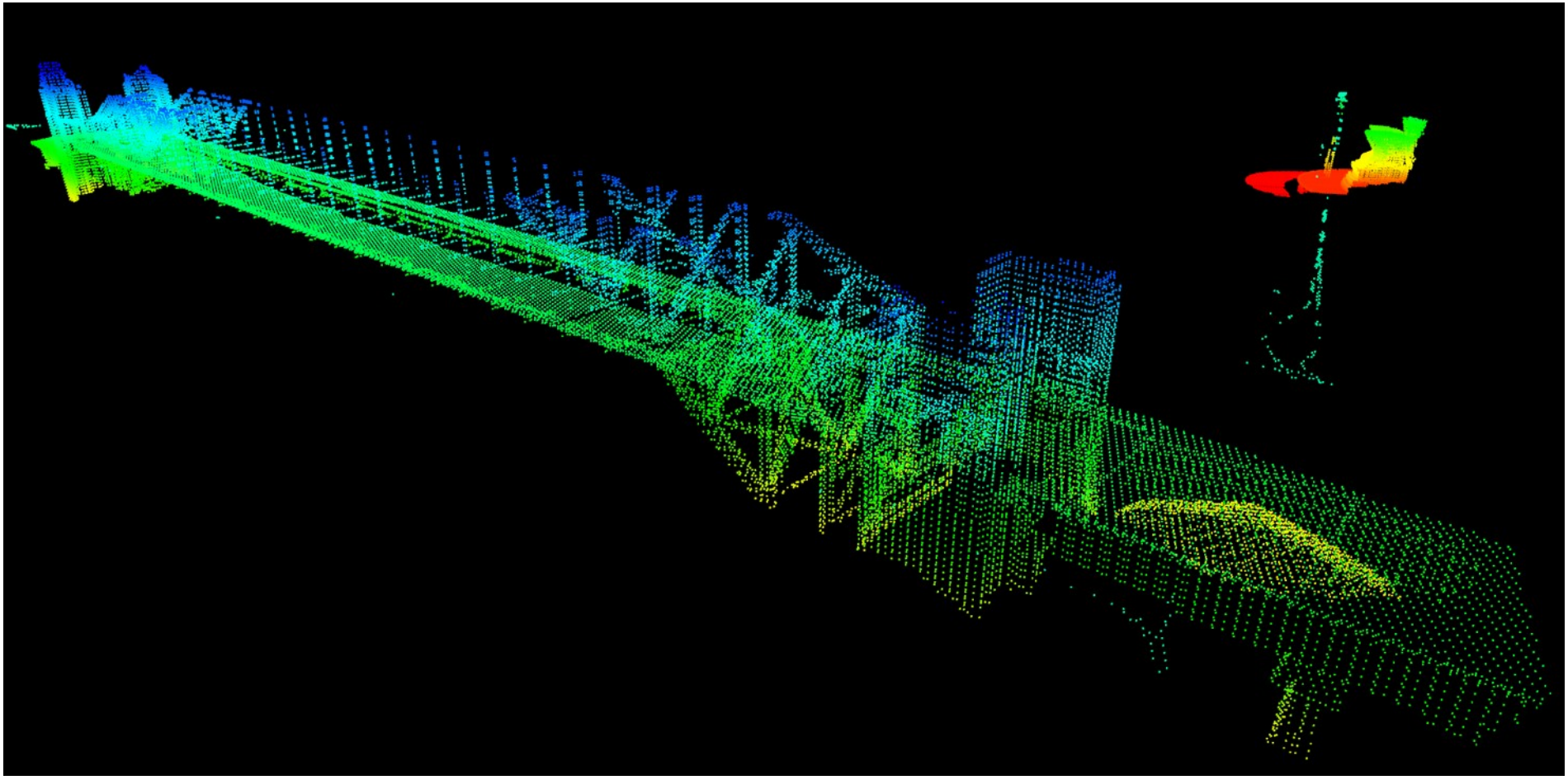
Includes all major structures



Mission start is at the Opera House

QGroundControl multi-vehicle mission launch

Result: Quad-tree assembled from five-drone team



Conclusions

- We have developed the tools to enable high-fidelity simulation of a team of lidar-equipped drones for our decentralised ISR project
- Our MOOS-IvP – PX4 controller integration should enable MOOS-IvP to be used for air and cross-domain autonomous systems (real and virtual)
- We have also demonstrated that MOOS can handle the high data rates associated with data-rich multi-vehicle simulations
- Our approach to GPU-accelerated sensor simulation is scalable and has been adapted to lidar, high-frequency acoustics and radar
- Although we have used MORSE as our target simulator, the same approach should work with other simulators in future

Perhaps a question for everybody: **Where to next...?**

Just in case we need another logo...