Norwegian University of Science and Technology (NTNU)

NTNU-FFI Cruise 2017

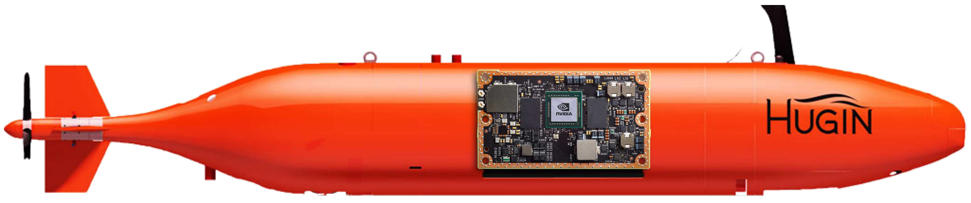
Hugin Autonomy Integration (DUNE, T-REX)

Øystein Sture (NTNU/AURLAB)

Martin Syre Wiig (NTNU/FFI)

Trygve Olav Fossum (NTNU/AURLAB)

The following report describes the integration of a dedicated autonomy co-processor on FFI Hugin HUS. The integration was experimentally validated during a joint FFI-NTNU cruise in November 2017. The cruise in its entirety is described in a separate report. Martin Syre Wiig assisted in testing, simulating and debugging the implementation. Trygve Olav Fossum assisted in the integration of T-REX.



Motivation

Hugin is equipped with a dedicated payload processor, where plugins can be loaded to perform remote control tasks. Its processing time is however shared with other functions such as data logging and sensor control. Autonomy on AUVs can require interpretation of spatial and/or temporal data, which can be computationally intensive. This can interfere with the other running tasks on the payload processor, or not receive high enough priority to perform the autonomy tasks successfully. A co-processor dedicated to autonomy avoids these issues, and allows for algorithms to be designed and tuned with a fixed processing power in mind.

The AUVs operated by NTNU are currently based around two platforms, Kongsberg/Hydroid REMUS 100 and OceanScan LAUVs (Sousa et al., 2012). Additionally, we have joint research cruises with FFI and their Kongsberg Hugin 1000. Although our REMUS 100 and Hugin both features the Hugin SDK interface, their versions (7.5 and 8.7) are not compatible. To make our research in autonomy agnostic across these platforms, we need an abstraction layer around data acquisition and remote control. Using the protocols defined by DUNE (Pinto et al., 2012) is a natural choice given that it already runs on our two LAUVs (Fridtjof and Harald). A deliberative executive autonomy module for goal directed control, T-REX (Py et al., 2010; Rajan and Py, 2012; Rajan et al., 2012), can then also be integrated by re-using its DUNE bindings.

Summary

Between the 10th to 16th of November, an autonomy co-processor running DUNE was utilized on-board Hugin to address the aforementioned issues. The co-processor was a Nvidia Jetson TX2, an embedded computer which has a good ratio between performance, physical size and power consumption. This computer runs DUNE and interacts with Hugin through a plugin on the Hugin payload processor over UDP sockets. In DUNE, this interaction is handled transparently, because Hugin acts as the navigational filter and path controller. The tasks corresponding to these functions in DUNE are disabled. A benefit of this clear separation is that no changes needs to be made in DUNE outside of a custom configuration file.

The Hugin-DUNE bridge currently supports sensor data from the multi-beam, side-scan sonar, DVL, navigation filter, CTD and altimeter. The remote control inputs are accepted in form of waypoints, speed and altitude/depth commands. Attitude control and direct control of Hugin can be exposed through the same interface without major modifications, but is not implemented at this time. This choice was deliberate, as these control schemes requires more validation ahead of use and was not needed for the planned experiments.

Basic operation and remote control was validated during a deployment at Tautra, where the primary objective was HiSAS/EM2040 data collection. Some bugs were discovered and resolved before the next deployment. The approach of adding small-scale tests onto deployments with other goals worked well and reduced downtime due to launch and recovery. The implementation was then validated by two autonomy experiments, one using DUNE directly and the other using T-REX as an autonomy layer.

The T-REX autonomy deployment was done in Korsfjorden. In this location, fresh water runoff, tidal, and the Coriolis effect creates a number of highly dynamic processes in the water column. The goal was to test a data-driven sampling algorithm, aiming to search and locate these phenomena in the region where they exhibit the greatest cumulative change across the Fjord. The first deployment failed due to a discrepancy between the configured acceptance distance in Hugin and DUNE respectively. A quick recovery and re-deployment enabled Hugin to cross the fjord several times, completing the experiment. The algorithm successfully changed its sampling location in response to the observed data, towards an area with more variation and scientific interest.

The DUNE autonomy experiment was performed at a dumping field at Agdenes, south of Brekstad. The goal was to perform an unsupervised segmentation of the acoustic backscatter response in the operation area and perform a camera survey for each seabed class discovered. The location was picked from previous HiSAS surveys in the area, which showed two types of sedimentation, possibly due to a submarine landslide. The camera lines were planned by maximizing the distance to the other classes, while staying within the operational boundaries. The test was performed twice, once at 10 meters altitude and again at 6 meters altitude. Both runs produced similar paths, with slight changes in the orientation of the camera lines.

Overall the integration has been successful. There are, however, some differences compared to the DUNE path controller. In some instances DUNE expects the AUV to loiter to attain a goal in the vertical plane after it has reached it in the horizontal plane. This is possible for Hugin by switching from waypoint to heading control, but is not implemented at this time. The current Hugin-DUNE integration instead sets a high value for the vertical waypoint tolerance such that it is accepted if reached in the horizontal plane. Care also needs to be taken to always ensure that Hugin has its next waypoint when needed, otherwise the remote control is terminated. This was ensured by increasing the horizontal acceptance radius in DUNE to exceed that of Hugin.

Hardware integration

The hardware payload consists of a Nvidia Jetson TX2 module mounted on a Connect Tech Orbitty expansion board. The Nvidia module itself only provides the CPU and GPU, with no I/O or network capability. An expansion board is necessary to provide power and Ethernet. The module was mounted in the dry section of Hugin. The power was supplied from a payload relay, which could be switched on and off independently of other payloads.

The Nvidia module can operate at several frequencies and with a different amount of cores enabled depending on the desired performance, power consumption or thermal constraints. The processing power and memory greatly exceeds other embedded boards in a comparable form factor, and additionally has a capable GPU which can be used directly through CUDA or CUDA-enabled frameworks such as OpenCV and TensorFlow. During our experiments, the module operated in the Max-P Core-All mode, where all CPU cores and GPU are enabled, but not clocked at their highest possible frequency (1.4/1.12 GHz vs 2.0/1.3 GHz CPU/GPU). The maximum clocks were not used because the module was mounted without a heat sink. The temperatures seemed to stay within 50 °C to 60 °C under load (CPU).

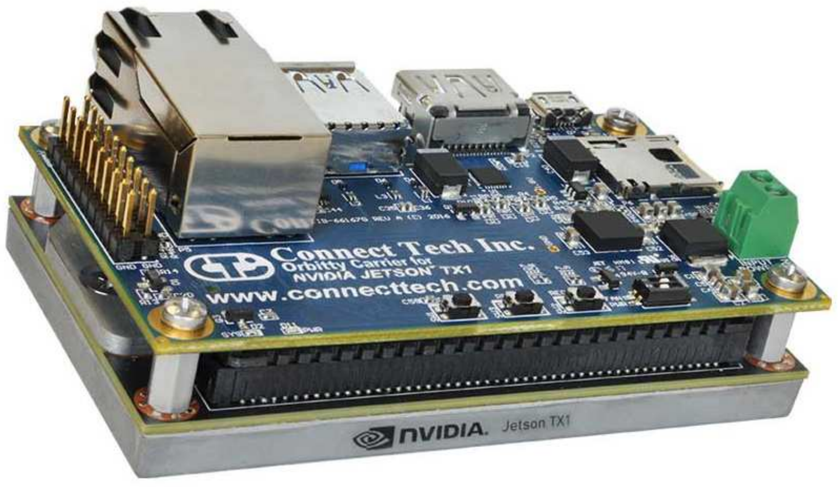


Figure 1: Nvidia Jetson TX1 with Connect Tech Orbitty expansion board

Table 1: Nvidia Jetson TX2 & Orbitty specifications

GPU	NVIDIA Pascal™, 256 CUDA cores
CPU 1	HMP Dual Denver 2/2 MB L2
CPU 2	ARM Quad A57/2 MB L2
Memory	8 GB 128bit LPDDR4, 59.7 GB/s
Internal storage	32 GB eMMC
External storage	128 GB microSDXC UHS-I U3 / Class10
Dimensions	87 mm × 50 mm × 30 mm
Weight	75 g
I/O	Gigabit Ethernet, USB 3.0, HDMI, GPIO, I2C, UART
Power req.	9 V to 14 V nominal
	Orbitty - 2 W
	TX2 - 6.5 W to 15 W (depends on enabled cores/frequency)

A 128 GB SD-card was added for the purpose of logging. The buffering was increased slightly to reduce the number of write cycles.

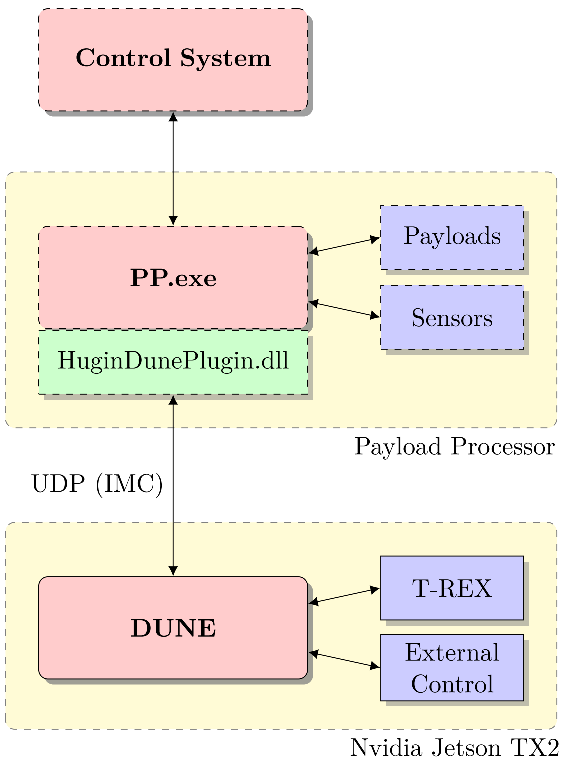


Figure 2: Diagram of components and processes

Software integration

The software integration consists of a plugin running on the payload processor (PP), which communicates with Hugin through the Hugin SDK. The sensor and vehicle data is serialized and sent to DUNE over UDP, and the reverse is performed for remote control commands from DUNE. The data is serialized according to the inter-module communication protocol (IMC) (Pinto et al., 2013), which DUNE uses to exchange data both internally and across systems. A table of messages that are exchanged between Hugin SDK and IMC can be seen in table 2 and table 3. The IMC messages are documented at the LSTS website (https://lsts.pt/docs/imc/master).

The plugin (HuginDunePlugin.dll) can operate in two modes, *powered on* and *activated*, controlled by the operators. When the plugin is *powered on*, the Jetson module starts up and the payload plugin (Hugin SDK) starts sending sensor data to DUNE. When the payload plugin is activated, remote control commands from DUNE are accepted. This distinction is also made in DUNE by enabling and disabling control loops. Control loops designates which mode the low level controllers are operating in, e.g. path control, altitude control, pitch control or a combination of multiple modes. When the plugin is *powered on* but *inactive*, the control loops are set to *non-overridable external control*, meaning that DUNE control is disabled. Upon activation these flags are unset, allowing DUNE controllers to take over. Currently only path control (waypoint mode) is implemented, but attitude, heading and low level control can be exposed with minor modifications.

When the plugin is *activated* and receives a control loops message which enables path control, a remote control token is requested from Hugin SDK. The Hugin SDK must receive a *keep-alive* signal on minimum a 1 Hz timer for this token to remain valid. The *keep-alive* signal is only sent if a *heartbeat* message has been received from DUNE within the last 10 seconds. The heartbeat message from DUNE is not used to set the *keep-alive* directly, due to the unpredictable nature of networking. The token is released automatically if the control loops are disabled, if DUNE enters an error mode (vehicle state), or an *abort* message is received through IMC. The operators can at any point disable remote control by deactivating the plugin, without powering down the Jetson TX2 module.

Table 2: Outgoing messages (Hugin → DUNE)

Hugin SDK API	Outgoing IMC messages
VehicleData	Temperature, Pressure, Conductivity, SoundSpeed, Distance (altimeter), Salinity, Depth
CBathyData	SonarData
CSideScanData	SonarData
NavigationSolutionData	EstimatedState, Acceleration, NavigationUncertainty
RemoteControlData	PathControlState, ControlLoops, VehicleCommand
DopplerLogData	Distance, GroundVelocity, WaterVelocity
pluginCommand	UamRxFrame (acoms), SetEntityParameter (T-REX on/off)
newMissionLine	CustomManeuver

Table 3: Incoming messages (DUNE → Hugin)

IMC Message	Hugin SDK API
ControlLoops	RemoteControl, SpeedMode, DepthMode, GuidanceMode
DesiredPath	CurrentWaypoint, NextWaypoint, Depth, Altitude, Speed
DesiredSpeed	Speed
DesiredZ	Altitude, Depth
Heartbeat	KeepAlive
VehicleState	RemoteControl
Abort	RemoteControl

When DUNE operates using *path control*, the maneuvers (e.g *goto, lawnmower, yoyo*) sends waypoints to the path controller using DesiredPath messages. Different behaviors can be triggered for Hugin based on the given flags in this message.

- If the *START* flag is given, the message contains both a starting waypoint and end waypoint. Hugin will in this case perform an outside turn both when entering the line and exiting the line. This is useful for camera surveys where the line in its entirety is to be surveyed, and not cut short by a premature turn.

- If the *DIRECT* flag is given, Hugin immediately moves from its current position towards the location by setting the current waypoint. The current T-REX implementation in DUNE uses this mode for its waypoints.

- If neither of the above are set, the waypoints that are received are set as the next waypoint. This causes Hugin to move through the given waypoints, cutting the corners with inside turns. This mode provides smooth path trajectories and should be preferred. Regular DUNE plans and maneuvers use this mode for setting waypoints.

The current state of the path controller is reported back to DUNE using PathControlState messages. When Hugin is within the waypoint acceptance radius of its current destination, a flag is set in this message. This triggers DUNE to dispatch the next waypoint. The distance at which this occurs is set in the configuration file of the plugin (HuginDunePlugin.ini), and should exceed the acceptance radius of Hugin to ensure that it always has a waypoint. If the next waypoint is not available when required, Hugin terminates remote control and resumes normal operation (i.e. the previous plan). The acceptance radius of Hugin varies according to the angle of its current and next waypoint up to a value determined by the maximum turn radius. During these experiments, the maximum turn radius was configured at 75 m, and the plugin radius at 100 m. The proximity to the goal is calculated in the plugin as the sum of the distance from the current position to the first waypoint and from that waypoint to the next, should one be queued up. This allows Hugin to utilize queued waypoints, as used in normal DUNE operation, but still be strict enough to allow the DIRECT mode to function.

Communication between DUNE and the topside operator station is currently limited to payload plugin commands, sent through acoustic or over a radio link. These commands were used to enable/disable the different experiments (i.e. activate T-REX or activate DUNE plan), so that they could be performed during the same dive. Commands are translated into generic acoustic messages in DUNE, and requires the code associated with the experiment to implement an action in response to the message. Communication from DUNE to topside is currently not implemented, but can be exposed through plugin status messages and a payload data interface. Generic payload data must be parsed by a plugin in the operator console or forwarded to another program. The latter is likely the better choice, as no integration with the operator console is necessary.

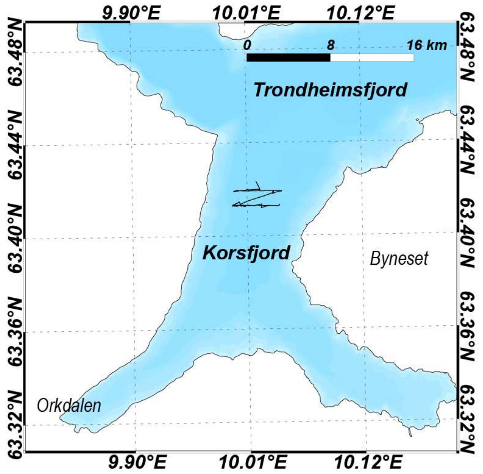
There are multiple layers of safety in the implementation. When the plugin is not active, both DUNE and the plugin will deny any attempts to enable remote control (software guards). If this for some reason should fail, the plugin can be powered down, cutting power to the co-processor (hardware guard) and shutting down the plugin. This in turn should cause the remote control token to be released within a second, as the keep-alive will stopped being sent. This does require an action by the operator however, and a radio link or acoustic link must therefore be available.

Additional safety guards should therefore be in place if communications are lost or Hugin exits the operational area. DUNE supports a set of behaviors if the communications are lost for a predefined amount of time (e.g keep station, go to launch, go to last comms). All these behaviors force the AUV to the surface. At the moment these are not functional, since remote control is disabled when the final waypoint is reached. Another DUNE specific safety feature is an operational area. This causes DUNE to enter an error mode if a set of vertical or horizontal bounds are exceeded. The remote control token is dropped on error mode, but if Hugin then returns inside the operational area, DUNE will resume normal operation and external code can once again take over. For these reasons, the communication loss timeout and operational limit supervisor were disabled. The DUNE behavior can be modified to work within these limitations, but it makes more sense to implement such features on a lower level (e.g in the remote control interface or the payload plugin).

Experimental validation

Adaptive Detection and Tracking of Sub-Mesoscale Processes (T-REX), Korsfjorden
Trygve Olav Fossum

T-REX is an onboard artificial intelligent agent architecture and has its origin from Monterey Bay Aquarium Research Institute (MBARI) and NASA. It is capable of automated planning (synthesized in-situ) and adaptive execution. One benefit of T-REX architecture is that it can interleave tasks that do not necessarily follow a sequential execution. This is important for long-term autonomy, where one may need to weigh the need to perform certain actions (e.g obtain GPS fix) against time spent sampling. A DUNE interface exists for T-REX, developed by José Pinto (LSTS, DUNE) and Frederic Py (T-REX), and only minor changes had to be made to the plugin to accommodate it onto the Hugin-DUNE system.



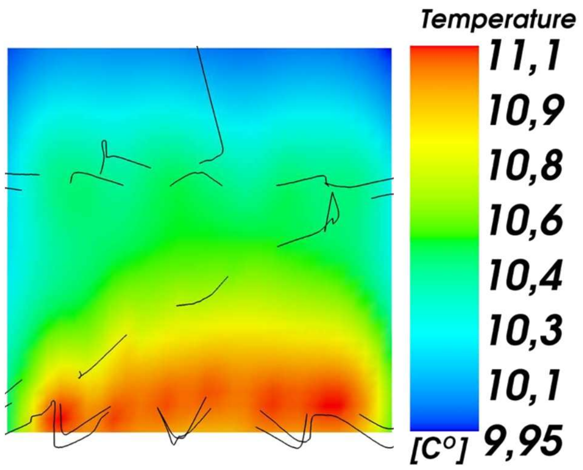


Figure 3: T-REX operation area with the mission track (left). Interpolated temperature plot at 42m depth (horizontal slice) with track lines moving through the layer (right).

The Korsfjord lies in the southwestern corner of the inner Trondheimsfjord and has a number of interesting characteristics. There are two major rivers (Gaula and Orkla) that deposit large amounts of freshwater into the fjord. This freshwater creates plumes that spread and interacts with tidal currents and are steered by the Coriolis force, as well as internal waves. The resulting interaction between these processes creates a number of time-varying structures on the surface as well as internally in the water column. The non-stationary behavior of these features suggest adaptive and data-driven approaches using autonomous underwater vehicles (AUVs) should be considered, as they can capitalize on both *in-situ* and *posterior* knowledge.

The main purpose of the experiment was to locate cross-sections of the fjord that had high level of change/gradients in temperature and salinity. The algorithm starts at the outlet of the fjord and traverses inward. After reaching/finishing a cross-section the algorithm stops to evaluate the measurements so far. There are two possible outcomes, repeat the same crossing or continue inwards (if no change is detected). This results in a data-driven sampling method where the AUV itself decides the sampling location based on the *in-situ* data.

The adaptive mission was initiated at 13:04 (T-REX activation on Hugin plugin), and finished at 14:50. Studying figure 3, Hugin approaches the starting point (small line from north to south at the top of the path) and then continue to traverse the fjord once. Once arriving at the surface at the eastern end of Korsfjorden (upper left corner) the AUV evaluates the data, and arrives at the conclusion that there is no particular change across the fjord at this latitude. Consequently, the AUV continues further inward, still recording data. On the journey inward, the AUV detects a change in temperature across a number of depths, prompting the AUV to turn and cross the fjord at this location (bottom left corner). The AUV crosses a large body of saline and warm water at this latitude, which is deemed interesting enough for the AUV to actually cross this body twice before finishing.

Backscatter Camera Survey (DUNE), Brekstad

Øystein Sture

Gathering multi-beam data from an AUV allows for near-seabed camera lines planned in-situ, which can reduce the need for ship-based sampling or deployment of remotely operated vehicles (ROV) to verify the seabed type. This experiment constructed a DUNE plan in-situ based on an unsupervised segmentation of collected multi-beam backscatter responses. The multi-beam survey was performed in a predefined area which was chosen based on HiSAS imagery from 2013. An interface between two seabed types was visible in the side-scan data, possible due to a submarine landslide.

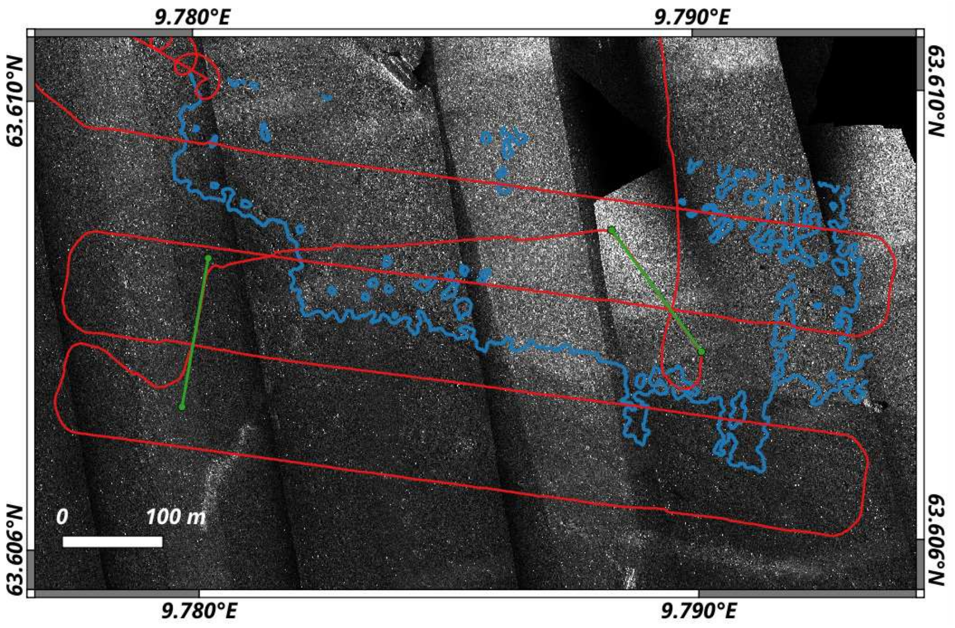


Figure 4: Side-scan sonar data from a previous survey (2013) with the paths and segmentation results from the new experiment as an overlay. The presence of a boundary is clearly visible, but might have shifted around due to currents. The boundary between the two classes are shown in blue, planned camera lines in green and the path of the AUV in red.

The backscatter returns were normalized based the mean response for a discrete set of incidence angles. This simultaneously removes the variations in intensity due to the angle of incidence and the sensor characteristics. The desired result is intensities which are comparable across the full swath for the same seabed type, thus suitable for segmentation. On completion of the pre-planned survey in the target area, an unsupervised segmentation was initiated by sending a command through the acoustic modem.

The segmentation is performed by modeling the probability distribution of the normalized backscatter response as a hidden Markov random field (HMRF) with a Gaussian likelihood model (i.e observation model). This model accounts for seabed types being characterized by different Gaussian distributions, but also integrates the notion that a location is more likely to belong to the same class as its neighbours. The labeling of the field and the parameters of the likelihood model are iteratively updated by alternating between optimizing over a fixed labeling and a fixed likelihood model.

One camera transect per class was planned by weighting the distance transform between classes while staying within the operational area. The segments were joined together by minimizing the distance traveled. Due to the small number of nodes, this was solved through brute force, but approximations to the traveling salesperson problem can be utilized for the general case. An underwater hyperspectral imager (UHI) with an internally mounted RGB was used for the images. The altitude reference was set at 6 m, which proved difficult for the RGB camera in these water conditions. The UHI, while not providing stellar images, indicated that the second camera line had a slightly elevated reflectance relative to the first line, but the same spectral signature.

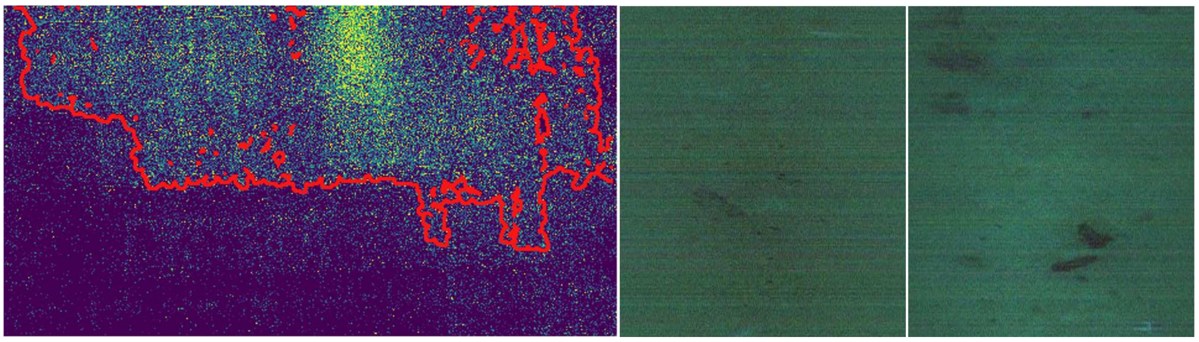


Figure 5: Normalized backscatter response on the left and normalized pseudo-RGB projections of camera line 1 and 2 respectively on the middle/right.

Integration Results

An issue with the hardware-in-the loop simulation on Hugin prevented full testing of the integration ahead of deployment. The issue seems to be resolved in version 8.9 of the Hugin OS, and was circumvented by running a simulated control processor (CP) at version 8.9, but using the Hugin payload processor (PP) at version 8.7. Having a simulator available is essential to validate ahead of deployment, and is something that should be verified to be in working state before future autonomy experiments. Ideally, NTNU should also have a software simulator available either permanently or in the months leading up to such a deployment. Tests of the hardware integration were tacked onto other deployments during the first two days, which helped resolving bugs before the experimental validation.

The first T-REX deployment failed due to a discrepancy between the configured maximum turn radius in Hugin (75 m) and waypoint acceptance radius in DUNE (60 m). This caused the remote control to terminate during the first deployment, as Hugin did not receive its next waypoint in time. When this occurs, Hugin resumes the next line of its pre-configured plan. The decision was made to recover Hugin and re-launch, as this occurred early in the mission. The next deployment completed all its goals successfully. The problem was not discovered during simulation, as the simulator had a different configuration than Hugin.

The second experiment was successfully executed twice during surveys in the Brekstad area. During one deployment however, the experiment failed due to payload processor (PP) rebooting. When this occurs, the power for the external computer is cut. The cause seems to be related to another payload plugin misbehaving. DUNE was set to auto-start on boot as a response to this, as it was started manually before. The payload processor is considered stable, but there is no reason to not run DUNE and external autonomy scripts on boot. This means that an experiment can be restarted by rebooting over the acoustic link.

Improvements and future direction

The following section lists possible improvements for the Hugin-DUNE integration in the future

- A payload plugin in Hugin SDK can report its status to the payload processor. This status is sent to the operators. The plugin status is currently set to idle at all times. This can be changed depending on the status of the co-processor/DUNE (e.g OperationError on communication loss with DUNE).
- Not all payload data is available through the SDK interface, e.g turbidity measurements, magnetometer, full EM2040 datagrams and HiSAS processed side-scan data. These are

written to files either continuously or on a line-by-line basis. Making these available to the co-processor can be important for future applications using these sensors.

- Certain operations in DUNE assume that it can move in the vertical plane through a elevator maneuver (loiter maneuver with depth change). This can be implemented in the plugin, but requires a change of operation mode on Hugin (heading/pitch control).
- When the plugin is powered down, the plugin exits cleanly but the power is cut abruptly for the co-processor. A clean shutdown should be implemented.
- There is currently no time synchronization between Hugin and the Jetson TX2. This can be resolved with NTP or another time synchronization scheme.
- It can be convenient to implement two-way communications over the acoustic link. Currently hex-commands can be sent from the operator console, but the reverse is not possible. Complex data structures must be interpreted topside, either through a GUI plugin in the operator console or forwarding the message over Ethernet.
- The operators are in full control of Hugin provided the acoustic link is maintained. For certain experiments, it may be desirable to release the remote control or assume a predefined behavior if Hugin is out of contact for a given time.
- For this cruise, Hugin was not equipped with a WiFi connection. This is practical as one can have access to the co-processor in the surface without performing a full recovery.

The future direction of this integration is not necessarily to polish the integration itself, but rather use it for developments in autonomy. The ability to test algorithms on an AUV with state of the art sensors and a high depth rating enables us to move in directions which would otherwise be out of reach. The current implementation has close to a full DUNE integration, which means that Hugin can also coordinate with other vehicles running DUNE.

Conclusion

The overall goals of the integration has been met. DUNE has been integrated on an autonomy co-processor which interfaces with a plugin on the payload processor. The implemented plugin appears to be stable, and allows Hugin to be controlled using DUNE maneuvers. Algorithms tested on DUNE can be therefore be transferred to Hugin without making major modifications, as the interface and behavior is compatible. DUNE also runs in a separate thread/process, which frees the autonomy code from considering concurrency. Modules with an existing DUNE integration, such as T-REX, works with minor or no modifications. The operational safety of Hugin is maintained provided that the operators are in contact via the acoustic link. A predefined behavior on timeout of the acoustic link can be added in the future as an extra layer of security.

References

Pinto, J., Calado, P., Braga, J., Dias, P., Martins, R., Marques, E., and Sousa (2012). Implementation of a control architecture for networked vehicle systems. *IFAC Proceedings Volumes*, 45(5):100–105.

Pinto, J., Martins, P. S. D. R., Fortuna, J., Marques, E., and Sousa, J. (2013). The LSTS toolchain for networked vehicle systems. In *MTS/IEEE Oceans*, pages 1–9. IEEE.

Py, F., Rajan, K., and McGann, C. (2010). A Systematic Agent Framework for Situated Autonomous Systems. In *9th International Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, Toronto, Canada.

Rajan, K. and Py, F. (2012). T-REX: Partitioned Inference for AUV Mission Control. In Roberts, G. N. and Sutton, R., editors, *Further Advances in Unmanned Marine Vehicles*. The Institution of Engineering and Technology (IET).

Rajan, K., Py, F., and Berreiro, J. (2012). Towards Deliberative Control in Marine Robotics. In Seto, M., editor, *Marine Robot Autonomy*. Springer Verlag.

Sousa, A., Madureira, L., Coelho, J., Pinto, J., Pereira, J., Sousa, J., and Dias, P. (2012). LAUV: The man-portable autonomous underwater vehicle. In *Navigation, Guidance and Control of Underwater Vehicles*, volume 3, pages 268–274.

Appendix: Nvidia Jetson TX2 setup

The Nvidia Jetson runs Linux 4 Tegra (L4T), based on Ubuntu ARM. During these experiments L4T version 28.1 was used. During initial setup, L4T must be flashed to the Jetson using JetPack. To enable additional features of the Orbitty expansion board, some modifications must be made to the kernel before flashing. Packages can be installed from the normal Ubuntu ARM repositories.

A static IP must be set to have network connectivity on Hugin. Before changing from dhcp to static IP the interface (eth0) must be shut down, otherwise the dhclient does not correctly shut down (stackexchange). It is best to resolve this before mounting the Jetson in Hugin.

```
# Shut down ethernet interface
sudo ifdown eth0

#
# Modify /etc/network/interfaces
#

# Bring back ethernet interface
sudo ifup eth0
```

Set a static IP by adding the following to */etc/network/interfaces*.

```
# /etc/network/interfaces

# Include files from /etc/network/interfaces.d
source-directory /etc/network/interfaces.d

auto eth0
    iface eth0 inet static
    address 192.168.99.188
    netmask 255.255.255.0
    post-up route add -net 224.0.0.0 netmask 224.0.0.0 eth0

auto lo
    iface lo inet loopback
```

This also sets up an explicit route to allow multicast, through a post-up action. Multicast is used for DUNE system discovery. The communication between DUNE and the Hugin SDK plugin is not dependent on multicast, but is necessary to communicate with Neptus and dynamic nodes. The reason the multicast route needs to be explicitly added is that there is no default gateway. A gateway is responsible for routing traffic when no direct route exists (e.g. outside the subnet). A virtual network is therefore established by adding the route to the routing table.

If the interface is not shut down before adding the static IP, the dhclient will not register that it should not manage the connection. This causes the ethernet connection to disconnect after a certain time (default 5 min). The timeout value can be set in */etc/dhcp/dhclient.conf*. The problem was not reproducible using a router, even without internet, and only occured after integration on Hugin. This is possibly related to the lack of a default gateway.