An Overview of the MOOS-IvP Middleware

Abstract for the SHARC 2016 conference

Simon Rohou
ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, France, Brest
simon.rohou@gmail.com

Abstract

MOOS-IvP is a set of open source modules for providing autonomy on robotic platforms, in particular autonomous marine vehicles [1]. It is composed of open-source projects dedicated to robot software architecture and autonomy tools, namely: the MOOS middleware [3], a cross platform middleware for robotics research and the IvP part based on MOOS and dedicated to a set of tools implemented to form a full marine autonomy suite known as MOOS-IvP. Mainly developed by the Oxford University for the MOOS part and the MIT LAMSS Laboratory for the IvP part, this software is nowadays widely used in the community of marine robotics.

1 Introduction

Today's robotic systems are becoming increasingly complex while being more and more connected with each other. The use of a trusted software base is essential to ensure reliable behaviors of a robot during complex missions and simplify inter-processes communications. Such software tools already exist and are known as *middlewares*.

In the field of robotics, a middleware is a software setting up a communication network between several *processes* – also called *applications*. It allows to break down the whole software part of a robot into several and well identified processes. Theses processes can be launched simultaneously or sequentially at the beginning of a mission.

Hence, the main advantages of a middleware are:

- to clearly separate the processes and thus clarify work distribution inside a development team ;
- to easily run processes in parallel and thus benefit from available computer's cores ;
- to distribute the processes over several machines connected to a common server ;
- to playback missions, because all variables updates can easily be logged ;
- to reuse already implemented applications meeting the needs.

Popular middlewares such as ROS, MOOS, YARP, MIRA, LCM, ... already provide these features. This presentation will give an overview of the MOOS-IvP middleware: the MOOS enhanced with the IvP software dedicated to autonomy features. Section 2 gives a brief presentation of the MOOS middleware. Section 3 presents the IvP part with the *Helm* and the behaviors framework. Section 4 concludes this abstract.

2 The MOOS

MOOS [3], a Mission Oriented Operating Suite, is a middleware based on a *publish-subscribe* architecture. Created by Paul Newman (Oxford University), MOOS is nowadays widely used by the research community especially in the field of marine robotics, embedded on Unmanned Surface Vehicles (USV) and Autonomous Underwater Vehicles (AUV). It can be used in C++ or in Python.

2.1 Publish-Subscribe architecture

This architecture depicts how the different processes communicate inside a given *community*. To understand this notion, let us take the example of an Autonomous Underwater Vehicle performing a wall following, see Fig. 1.

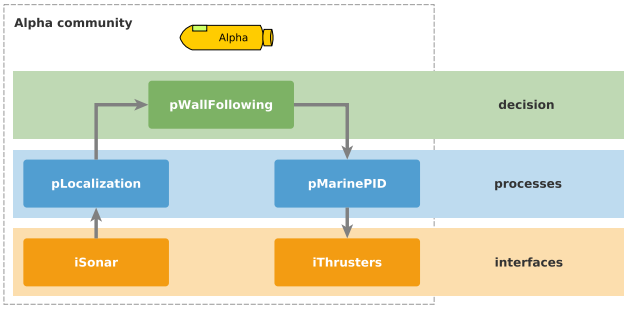

Figure 1: a community onboard of an AUV

The robot is equipped with a sonar and propelled by some thrusters. This represents the hardware part of the AUV and dedicated applications will be used as an *interface* between this hardware and the rest of the robot's software (see *interfaces*, Fig. 1), namely: `iSonar` and `iThrusters`. The processing of sonar data and the controller (PID) are pure *processes*: `pLocalization` and `pMarinePID`. The high-level dedicated to robot's behavior is held by `pWallFollowing`. It aims at deciding a direction based on the estimated localization.

These processes form a chain. In a publish-subscribe architecture, all the communications are centralized within a database. In the context of a MOOS

Community, we speak about a *MOOSApp* (for processes) and a *MOOSDB* (for the database). See Fig. 2.

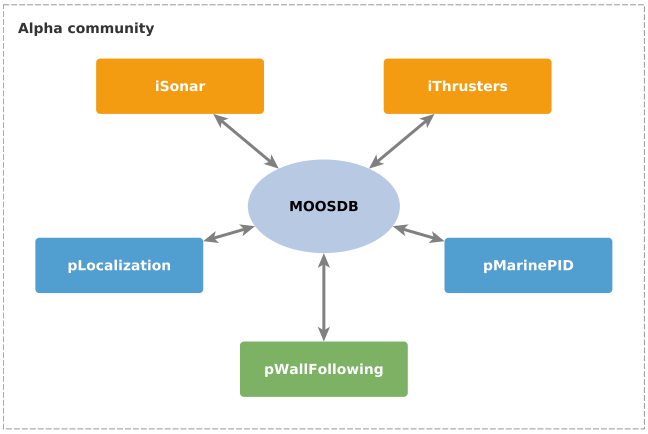

Figure 2: a publish-subscribe MOOS community

A Community, defined by a MOOSDB, is referenced by an IP address and a port. Some MOOSApps take place inside a community and exchange data through shared memory when running on a single machine or using UDP in the case of a distributed architecture.

When a MOOSApp `A` has to communicate a MOOSVar `c` to an other MOOSApp `B`, the following process is executed: `B` subscribes to possible changes on `c` in the MOOSDB. Then `A` publishes a new value for `c`, stored in the databse. Finally `B` receives a notification and can read `c`'s new value.

Thus, all communications occur through the MOOSDB, which is useful to log data exchanges. MOOS provides tools (dedicated MOOSApps) to log and playback these communications.

2.2 Implementation

The developer will implement a MOOSApp in C++ or Python, based on the inheritance of a MOOSApp class. The user will then configure the MOOSApp with a `.moos` file, specifying community's parameters (IP, port) and some custom options.

3 MOOS-IvP

The MOOS-IvP autonomy software [2], based on the MOOS middleware, is a C++ open source project providing a full marine autonomy suite, in particular: a behavior-based application for decision making, called the *helm*, and additional tools for simulation, viewing, inter-communities communications, marine controllers, etc. Created by the LAMSS[1] (MIT), this software is nowadays well used in the field of marine robotics and gathers a community of researchers and industries.

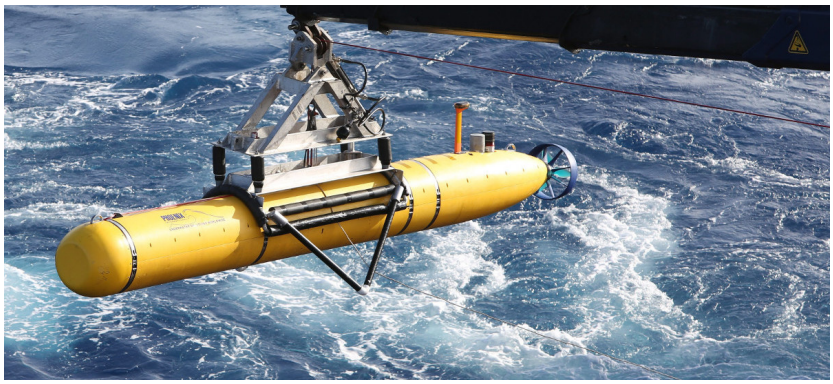

Figure 3: a Bluefin Autonomous Underwater Vehicle powered with MOOS-IvP

3.1 Autonomy with the IvP Helm

The autonomy, held by the *helm*, takes place in a dedicated MOOSApp called `pHelmIvP`, see Fig. 4.

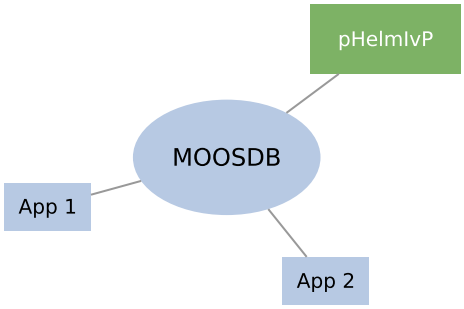

Figure 4: the *helm* taking place in `pHelmIvP`

The *helm* presents a behavior-based architecture in which several behaviors can be added according to mission's needs. During the mission and depending

[1]LAMSS: Laboratory for Autonomous Marine Sensing Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

on the context, some behaviors can be triggered while other can stay disabled. This ensures an appropriate answer to the environment's conditions.

Each behavior can be seen as a software library. Subscribing to some data (MOOSVars, *e.g.* the estimated position), the library will propose a behavior (MOOSVars for instructions, such as a desired speed or a bearing to follow). Several behaviors can run simultaneously. The strong point of the helm is that each behavior will only *propose* a solution: it is the solver's role to determine the final behavior, based on the computed options. Hence, behaviors are competing for influence of the robot, see Fig. 5.

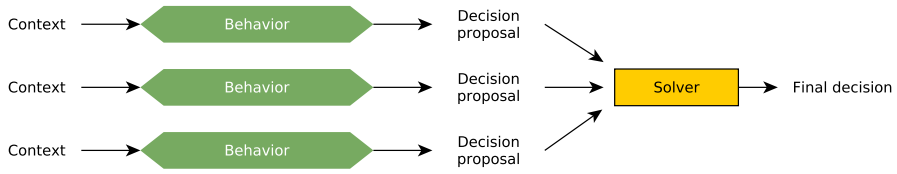

Figure 5: the *helm* presenting a behavior-based architecture

This gives the opportunity to have smooth and adaptive attitudes during a mission. For instance, if a robot has to follow a path while avoiding another robot, the *helm* will give an appropriate weight to each behavior: an increasingly strong one for the obstacle avoidance when the robot will approach the object and a constant one for the path following.

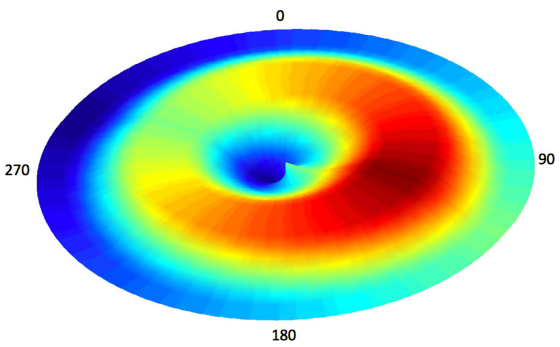

Figure 6: example of an *IvP function* used to propose a bearing decision (from 0° to 360°). A behavior outputs several IvP functions giving preferences over a variable domain such as the speed, the bearing, the depth. In the figure, the hottest a point is, the better its chances of selection by the solver, according to concurrent IvP functions.

3.2 Overview of already existing behaviors

Some behaviors are freely available:

- `Waypoint`: the robot will follow a defined path
- `AvoirCollision`: to avoid collisions between moving vehicles
- `PeriodicSurface`: when an AUV has to surface on a regular basis
- `ConstantDepth`: to maintain a desired depth
- ...

3.3 Additional provided tools

Some MOOSApps devoted to marine applications are provided with the *helm*:

- `uSimCurrent`: simulation of currents based on models or data files
- `uSimMarine`: simulation of a marine robot's physical behavior
- `uTimerScript`: to script the publication of MOOSVar custom values
- `pMarineViewer`: a graphical user interface to monitor robots and objects on a map, control MOOSVar, publish values, etc.
- ...

4 Conclusion

The MOOS-IvP software is one of the major tools in the field of marine robotics. Its autonomy framework provides a reliable behavior based architecture accompanied by ready-to-use behaviors. Other robotic fields could take advantage of this software.

The presentation will be illustrated with robotics examples such as the simulation of an autonomous boat performing a mission in the *Anse du Moulin Blanc* (Brest) or the concrete demonstration of a semi-autonomous boat on Polytechnique's lake, see Fig. 7.

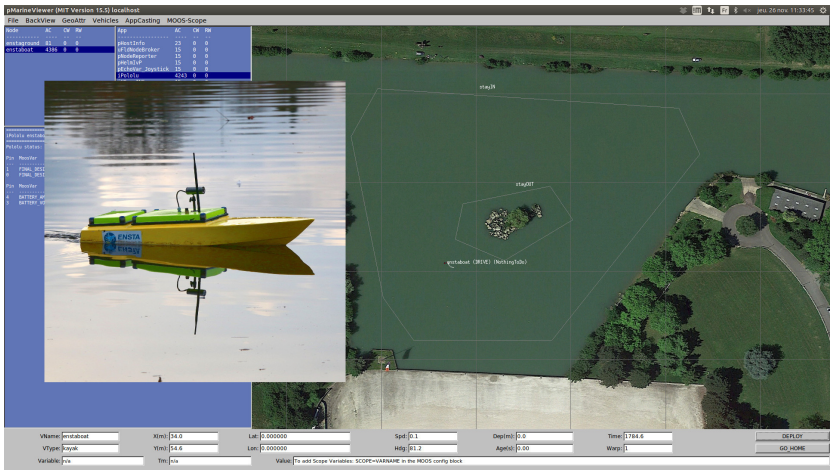

Figure 7: demonstration on Polytechnique's lake with ENSTA Bretagne's autonomous boat, powered by MOOS-IvP – the boat had to stay in its *viability kernel*: controlled by humans, the boat becomes autonomous when leaving its safety area (represented by two polygons) in order to come back to a safe state. A dedicated *behavior* has been implemented to manage this attitude. The behavior was only triggered when the viability kernel was about to be left.

References

[1] Michael R. Benjamin, Henrik Schmidt, Paul M. Newman, and John J. Leonard. MOOS-IvP: a set of open source modules for providing autonomy on robotic platforms, in particular autonomous marine vehicles. `http://www.moos-ivp.org`.

[2] Michael R. Benjamin, Henrik Schmidt, Paul M. Newman, and John J. Leonard. Nested autonomy for unmanned marine vehicles with MOOS-IvP, 2010.

[3] Paul M. Newman. MOOS: Mission oriented operating suite, a cross platform middleware for robotics research. `http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php`.