APM(Robot): Towards a platform for meta-reasoning in robotic applications

Cédric Dinont and François Gaillard and Michaël Soulignac

Institut Supérieur de l'Electronique et du Numérique
41, Bd Vauban 59046 LILLE Cedex

{firstname.lastname}@isen.fr

Abstract: This paper describes the design of APM(Robot), a platform aiming to provide meta-reasoning capabilities to robotic agents. Meta-reasoning could facilitate integration tasks in robotic applications, involving heterogeneous modules or robotic systems. APM(Robot) is based on APM, an agent-oriented library providing generic services for communication and storage tasks. APM allows reifying communication concepts such as communication channels, languages and contacts. It also allows carrying out distributed and event-based data stores.

Keywords: Multi-agent systems, meta-reasoning, middleware for robotics, robotic simulations.

1 Introduction

Robotic applications involve many skills including software engineering, electronics or mechanics. Each skill is in charge of several modules, each one with its own expertise. If we only consider the software part, we could need algorithms from several domains: vision, localization, goal planning, path planning, motion planning, etc. To build one single homogeneous application, we need a platform able to integrate all those disparate reasoning modules. Moreover, each module may have limitations that prevent it to optimally behave in the global system. We think that meta-reasoning capabilities may bring significant improvements on the consistency of robotic applications that involve modules from different sources. Following Pitrat's point of view ([Pit90]), we think that efforts should be made to use meta-reasoning everywhere it is possible because it allows a system to adapt to many more situations than systems than do not use introspection. It is especially important in the robotics field where developments are too often ad hoc.

Next sections describe the foundations of a platform that can help achieve these goals. We first describe APM, a general purpose agent oriented communication and (meta-)reasoning library. APM reifies concepts like communication channels, communication languages and contacts. It also offers several data history management mechanisms. Then, we introduce APM(Robot), a robotic platform which uses the services of APM and proposes generic modules used in robotics on which we can apply meta-reasoning.

2 APM

APM (Agent Platform for Meta-reasoning) is the foundation on which we base our robotic platform. It currently provides support for two important concepts used in cognitive agents applications: communication and memory management. Our interpretation of these concepts is presented in the following sections.

2.1 Communication toolkit

Our communication toolkit is intended to be used in a large variety of applications. It is not a complete communication framework that can be directly used in application. It only provides generic concepts that can be extended or customized in a given distributed platform. The toolkit contains several functionalities that ease the development of a concrete middleware. To maximize its versatility, it is designed to allow most of its functionalities to be bypassed if it is useless or senseless in a given context.

We reify three main concepts in this toolkit: contacts, communication channels and communication languages. They are described in the following paragraphs.

2.1.1 Contacts

Basically, a contact can be seen as the way we represent the sender and the recipient of a message. We do not enforce any representation for contacts, as it can vary upon applications. Our contacts model is based on the human metaphor: a contact may be known with multiple names and we can send it a message using different physical communication links. Contacts information is kept in an address book where we put parameters used to init a communication with a given physical link.

2.1.2 Communication channels

A communication channel is the common place where communication tasks take place. It manages physical links between an entity of the system and other entities it exchanges messages with. Physical links can be of any type: socket, Bluetooth, I2C, email, etc. Communication languages are delinked from physical links. Thus, physical links can support any communication language.

Message reception is asynchronous: physical links manage the low level details of the communication, ask a communication language module to decode the incoming data stream and put the resulting message objects in a mailbox. The incoming messages queue can be accessed directly or can be processed by a dispatcher that routes messages to internal modules.

Messages can be sent synchronously or asynchronously. In the latter case, messages are enqueued in the communication channel which uses customizable policies to try to send them to the recipient. For instance, a policy can select the physical link that will be used and another will try to send the message X times with a delay between retries. Errors raised during this process generate messages that are placed in the mailbox.

2.1.3 Communication languages

We define a communication language upon the three following concepts:

- The signifiant of a word, i.e. the idea, the concept behind it.

- The signifier of a word, i.e. the way to represent it.
- The communication medium, i.e. a specific way to express words.

A communication language represents the links between signifiers and their signifiant, according to the Ferdinand de Saussure's model [dS16]. We define a communication process as the usage of a communication language through a given communication medium with the right marshaller. A marshaller provides the two following processes:

- A marshalling process of a word, which is the specific transformation of this word to express it on a given communication medium, like a serialization process.
- An unmarshalling process of a word expressed on a communication medium, which is the backward transformation of this word to its significant form, like a deserialization process.

The signifiers are specifically used here to identify signifiants during a communication process. A marshaller can also coordinate chained marshallers in order to create a communication process through multiple communication mediums.

2.2 Data store

A data store is the "intelligent storage" module provided by our platform, mixing concepts of producer-consumer and event-driven programming.

Presently, a data store can be seen as a -possibly distributed- blackboard. Several agents, or several modules within an agent can add data, and subscribe for data changes in their area of interest, in order to achieve a complex and common goal. Each area of interest is materialized through an entity called "data history".

In the future, data stores are aimed to include meta-reasoning capabilities on manipulated data.

2.2.1 Data histories

A data store manages data histories, which are containers for the successive values of a given data (simple or compound) upon time. We can attach meta-data to data histories. They can be used to find data histories in the data store or to reason about them.

Several policies can be associated to data histories:

- A policy that manages how values are deleted from the history when time passes. This is used to forget data when they are no longer needed.
- Software modules can register themselves to receive events generated by data histories (value added, value deleted, meta-data added, data-data deleted). Customizable policies are used to indicate how and when events are generated.

A data history can be seamlessly manipulated locally or remotely, thanks to a set of meta-data that describe the link between a fake local data history and a real remote data history.

2.2.2 Meta-reasoning

Meta-reasoning capabilities are not implemented yet. Our goal is to allow adding generic meta-data to data histories and reasoning modules manipulating them. By this way, data stores could be able to automatically identify the modules or agents which are appropriate to achieve a given goal, and organize the collaboration between them.

The data store we propose has similarities with self-organizing systems, such as modular robots [ArRS+07]. The main difference is that modules and/or agents could be strongly heterogeneous. This is an important feature because we want our platform to ease the integration and collaboration between heterogeneous AI and signal processing technics.

3 APM(Robot)

APM(Robot) constitutes the foundation of our robotic software platform. It provides general purpose modules that can be used in robotic applications. Entry points are given to implement low level parts that are specific to chosen robots.

3.1 Object versus agent oriented programming

Almost all robotic platforms are based on object oriented programming, like Player ([GVH03, CM05]), or service oriented programming, like Microsoft Robotics Studio. We think that these approaches, although allowing to write applications that can be executed on several robots without any change, are not suitable for our needs. Indeed, we want to reason about the specific capabilities of robots, which are hidden by object or services models. Cognitive agent paradigms are more appropriate to our point of view. If we reason on a declarative model of robots (their hardware components and software modules), we will be able to write high level programs that will automatically adapt their behavior to robots they run on.

3.2 Sensors manager

The sensors manager module has a description of the sensors connected to the robot. It schedules capture requests on every sensor and send generated data into the data store. The event system is then used to wake up the modules that consume sensors data. In the future, the sensors manager will be able to dynamically adapt sensors capture requests schedule to the real needs of the robot (see next paragraph).

3.3 Localization

We think that the meta-reasoning capabilities of our platform are useless for the integration of different localization algorithms in an application because data fusion from different localization modules does not usually give good results. APM(Robot) will only be used with fully integrated localization approaches like the ones that use Kalman filters [HA06] or particles systems [FTBD00, CMH+08]. Meta-reasoning will be used to analyze the usage made of the sensors data by the localization module and to provide the results to the sensors manager that will eventually change its capture policy.

3.4 Planning

Based on the top-down model established by [Rey99], we can distinguish three planning levels in robotic applications: goal planning, path planning and motion planning.

The interactions between these levels are the concern of hybrid motion planning ([Kha86], [LK01] or [BKV02]). It mixes both path planning and motion planning levels in order to create more efficient motion planning strategies. Hybrid motion planning especially aims the mix of the respect of physical constraints in motion planning and the efficiency of specific path planning. In order to automaticaly build overall motion planning strategies, we see these planning levels as planning modules. Then, we want to apply meta-reasoning on:

- The inputs and outputs of motion planning levels, i.e. which kind of data are exchanged between them.
- The inner reasoning, i.e. how the planning level works; we want to identify sub-reasoning specific to this planning level and use them as further planning modules.

Our goal is then to find the most efficient motion planning strategy regarding the situation of the robot by weaving the more appropriate planning modules.

3.5 Environment representation

As our platform is intended to integrate modules from different sources, we need to provide a common way to represent data shared between modules. The environment representation is an important part of this shared data. To achieve this goal, we cannot arbitrarily choose a data structure, because each module needs a specific representation. Thus, we propose a module that can manage multiple representations for every element of the environment. We provide a visualization module for the environment that allows to superimpose, thanks a system of layers, differents views of the same environment.

Our multiple representations model allows to observe, for instance:

- Several models of the environment: continuous, discrete (grid, visibility map), potentially with different granularities.
- Several views for the same entity: geometrical shape (polygon, circle, etc), picture, etc.
- For a same model and view: a set of hypotheses for an entity.

This last point allows representation, for instance, of different hypotheses for the state of the robot, resulting from different localization modules while comparing them on a particular situation before choosing the one that will be used in an application.

4 Conclusion and perspectives

Our still in progress platform called APM aims to gather reasoning upon its own heterogenous modules. APM will provide meta-reasoning on the capabilities of its support. Our first implementation is used in APM(Robots), which provides meta-reasoning about robots tasks: planning, sensors management or localization. We wish to reduce the gap between each modules: how they work and how they interact. Our next goal is to reason about the motions which provide information about environment places out of the perception radius of the robot. Then, we should be able to provide new robots languages, with the view to manage their motion policies.

References

[ArRS+07] Michael P. Ashley-rollman, Michael De Rosa, Siddhartha S. Srinivasa, Padmanabhan Pillai, Seth Copen Goldstein, and Jason Campbell. Declarative programming for modular robots, IROS 2007.

[BKV02] Oliver Brock, Oussama Khatib, and Sriram Viji. Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 388–393, 2002.

[CM05] Toby H. J. Collett and Bruce A. Macdonald. Player 2.0: Toward a practical robot programming framework. In *Proc. of the Australasian Conference on Robotics and Automation (ACRA)*, 2005.

[CMH+08] Guanghui Cen, N. Matsuhira, J. Hirokawa, H. Ogawa, and I. Hagiwara. Service robot localization using improved particle filter. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 2454 –2459, sept. 2008.

[dS16] Ferdinand de Saussure. *Cours de linguistique générale*. 1916.

[FTBD00] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. *Sequential Monte Carlo Methods in Practice*, chapter Particle filters for mobile robot localization, pages 470–498. Springer-Verlag, 2000.

[GVH03] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[HA06] N. Houshangi and F. Azizi. Mobile robot position determination using data integration of odometry and gyroscope. In *Automation Congress, 2006. WAC '06. World*, pages 1–8, july 2006.

[Kha86] O Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, 1986.

[LK01] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *International Journal of Robotics Research*, 20(9):729–752, September 2001.

[Pit90] Jacques Pitrat. *Métaconnaissance, Futur de l'Intelligence Artificielle*. Hermès, 1990.

[Rey99] Craig Reynolds. Steering behaviors for autonomous characters, 1999.