A Distributed User-Centered Approach For Control in Ambient Robotic

Nicolas Verstaevel, Christine Régis, Marie-Pierre Gleizes, Fabrice Robert

A Distributed User-Centered Approach For Control in Ambient Robotic

N. Verstaevel[a, b] verstaev@irit.fr

C. Régis[b] regis@irit.fr

M.P. Gleizes[b] gleizes@irit.fr

F. Robert[a] fabrice.robert@sogeti.com

[a]Sogeti High Tech,
3 Chemin de Laporte, Toulouse, France

[b]IRIT, Équipe SMAC,
Université Paul Sabatier, Toulouse, France

Abstract: Designing a controller to supervise an ambient application is a complex task. Any change in the system composition or end-users needs involves re-performing the whole design process. Giving to each device the ability to self-adapt to both end-users and system dynamic is then an interesting challenge. This article contributes to this challenge by proposing an approach named Extreme Sensitive Robotic where the design is not guided by finality but by the functionalities provided. One functionality is then seen as an autonomous system, which can self-adapt to what it perceives from its environment (including human activity). We present ALEX, the first system built upon the Extreme Sensitive paradigm, a multi-agent system that learns to control one functionality in interaction with its environment from demonstrations performed by an end-user. We study through an evolutive experimentation how the combination of Extreme Sensitive Robotic paradigm and ALEX eases the maintenance and evolution of ambient systems. New sensors and effectors can be dynamically integrated in the system without requiring any action on the pre-existing components.

***Keywords**: Distributed Architecture, Innovative Architecture, Human System Interactions, Control System, Internet of Things, Smart Devices, Adaptive Multi-Agent System*

1 INTRODUCTION

We are living at a time where technologies evolve every day. Intelligence, once restrained in personal computers, is now distributed in our environments under many forms. Those systems are *ambient* (Guivarch, 2012). Internet of things, wearable sensors, robotics, home automation, are illustrations of the ubiquitous computing revolution (Weiser, 1991). As software and hardware become ever more elaborated, intelligence is now embedded in objects. We have at our disposal libraries of various components realizing functions rather than objectives. For example, smart cameras can produce data from image recognition algorithms or every day object can play a role in the human-system interaction. Each of those components is autonomous and designed independently.

A robot for a particular application consists in the aggregation of the necessary components to satisfy its objectives. Those components could be part of the robot body or distributed in its environment. The collective of components has to interact and collaborate to perform an adequate global activity. The design of an intelligent system is then a matter of integration and a recurrent challenge is how to enable all those intelligent things to collaborate whereas they have an autonomous activity.

Those ambient systems are truly complex: a potentially huge number of heterogeneous devices evolves autonomously (including appearance or disappearance of devices) to provide services to its users (Perera, 2014). Designing an *ad hoc* controller supervising the whole activity involves having a lot of knowledge on the system dynamic. Any change in the system composition implies re-performing the whole design process meaning that sustainability of such system is a challenging task. Complexity is increased by the specific, multiple and often changing needs of end-users. Designers cannot make *a priori* a complete specification. Actually, the maintenance and evolution of an ambient system involves high costs, as it usually requires high knowledge and skills.

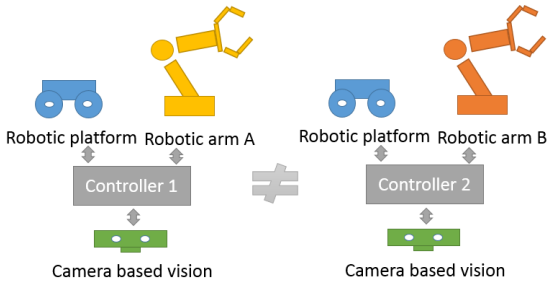

Figure 1: Example of integration problem: designing a unique controller for 3 robotic components is dependent of system composition.

One of the challenges is then to give to each device the ability to self-adapt to both system dynamic and end-user needs.

This paper contributes to this challenge by studying the benefits of using self-adaptive components in the design of robotic applications. The paper is organized as follows: First, we formulate the problem of integration of robotic. Secondly, we present the Extreme Sensitive Robotic paradigm, an innovative architecture to design ambient robotic applications. A scientific background is then provided to position the paper in regard with other scientific domains. Our main contribution, ALEX, is then presented in section 5. Section 6 proposes a use-case study of using ALEX in combination with the Extreme Sensitive Robotic paradigm from the viewpoint of a designer of a robotic application. Finally, we conclude with some perspectives.

2 DESIGNING A ROBOTIC SYSTEM: THE INTEGRATOR PROBLEM

The evolution of technologies, both in terms of hardware and software, makes available libraries of various components realizing functions rather than objectives. Internet of things (Perrera, 2014) is the perfect illustration of such a possibility. Designers of those systems have to aggregate different functions to build a system providing services to its users. Those functions are provided by electronic devices (basically by effectors). Each device exercises a control over a particular functionality. For example, a particular device can control the activation of an electric shutter and another one can control lights. Robots are part of those systems and propose a set of functionality, which can include mobility. A mobile robotic platform equipped with a robotic arm then provides two functionalities: the ability to move and the ability to grab objects. The integration of those different robotic components in order to provide service to humans is a complex task.

Let's consider the case of a designer who wants to integrate a robotic arm from one constructor and a mobile platform from another constructor, while using image processing algorithms from a third provider to perform a collecting task. The design of an *ad hoc* controller for such application is complex and requires a lot of expertise on each component, but also on its environment (which includes human activity). If for any reason, a component is replaced with another one (even if this new component provides the same functionality), the whole design process has to be performed again. This is time greedy and involves high cost of maintenance and evolution. However, the same system composition (one robotic arm, one camera based vision and a robotic platform) could be used in different kind of application. For example, one could want to use it for turning valves in a factory or the other for cleaning a room in a nuclear power plant. Each new application involves designing a new controller (figure 1).

A designer of such system would profits from a system capable of self-adapting to both the environment and users' needs without requiring reprogramming any system's component. This is the postulate made by *Extreme Sensitive Robotic*.

3 EXTREME SENSITIVE ROBOTIC: EXPECTATIONS

The Extreme Sensitive Robotic (XS Robotic) is an integrative approach of functions of perception, decision, action and interaction. It proposes a bottom-up approach focusing on functionality rather than a top-down approach focusing on objectives. Each function is an atomic part composing the micro-level of the system. A robot is then seen as the aggregation of the necessary functions to satisfy user's needs. Further, a group of robots or a whole ambient system has to be considered in the same way: a set of macro-functions (each robot) working in coordination.

The XS Robotic considers each robotic device in interaction with humans, other devices and the environment through sensors. Each device is autonomous which induces that the complexity of a robot (or a collective or robots) is not described explicitly or implicitly in it. Each device determines its own activity in interaction with its environment. The problem of integration then becomes a problem of adaptation. Each device has to adapt its behavior to its environment.

By applying the XS Robotic paradigm to the

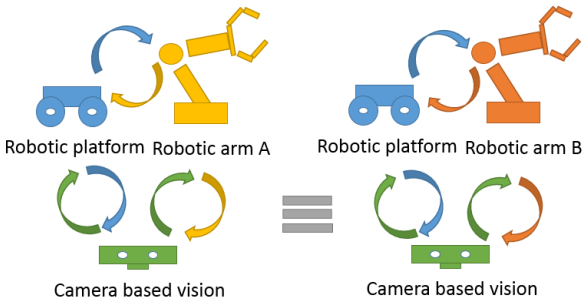

Figure 2: The same problem through the scope of XS Robotic. The two systems are an equivalent problem.

previously enounced problem, there is no difference between the two systems (Figure 2). Indeed, as each device is able to self-adapt to its environment, it will autonomously integrate any new device to its own activity.

To be truly effective, the XS Robotic needs generic algorithms allowing devices to self-adapt both to system's dynamic and humans. Those devices, which interact with their environment and their users, must have the capacity to automatically learn from this interaction and exploit this knowledge. But to be as natural as possible, the human-system interaction must rest on a process that does not need any expert knowledge. On contrary, it must provide a natural way for any kind of user to express their needs.

4 SCIENTIFIC BACKGROUND

On the previous section, authors present the *Extreme Sensitive Robotic* as an integrative approach resting on self-adaptation skills. *XS Robotic* deals with the ability to learn from interaction with the environment and users. However, the idea of making autonomous systems able to self-adapt and learn from their environment is not completely new. In fact it relies in the heart of informatics. Yet in the early 50's, Alan Turing (Turing 1950) states that "instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain". On this section we present concepts coming from robotic, cognitive science and artificial intelligence that attempt to build autonomous artificial systems with the ability to learn from interaction. For each domain, we point out main properties required for enabling *XS Robotic*.

More than sixty years after the dream of Alan Turing, robotic controllers are still handcrafted. Artificial intelligence failed to bring Turing's dream to life. Brooks explains that this failure may come from engineer's conceptualization of the world that may not be appropriate for artificial systems with a different sensory motor apparatus (Brooks, 1990). Due to the limits of introspection, the abstraction that a human would supposed to be appropriate to build a system may be completely different to what he is actually using. A metaphor that sums up Brook's idea would be that making abstraction of the world is like *observing the world through a keyhole instead of opening the door*, depriving the system of all the wealth that this world has to offer. To avoid this problem, Brooks proposes the *physically grounding hypothesis* that stipulates that interaction with the environment has to be the primary source of constraint for the design of intelligent system.

Pfeifer (Pfeifer, 2006) goes further by arguing that there is a strong relationship between the body and the mind. Pfeifer states that the traditional view of intelligence is that it is located inside the brain, or more generally inside the control system. However, he shows that studying the brain (or the control) alone does not allow to completely infer the behavior of the system. The brain needs a body to act, and the way the brain is embodied in the physical world may strongly influence the way it acts. This relation is called *embodiment*. Embodiment plays an important role in learning as what we can learn is strongly related to what we can do.

Zlatev and Balkenius (Zlatev, 2001) state that cognitive science community realizes that "true intelligence in natural and (possibly) artificial systems presupposes three crucial properties:

- The **embodiment** of the system
- Its **situatedness** in a physical and social environment
- A prolonged **epigenetic developmental process** through which increasingly more complex cognitive structures emerge in the system as a result of interactions with the physical and social environment

Cognitive sciences have a particular echo inside the artificial intelligence community and has inspired learning techniques. (Guerin, 2011) proposed an overview of artificial intelligence approaches trying to build programs that could develop their own knowledge and abilities through interaction with the world. The approaches inventoried by Guerin share the same conception of the learning process. They see learning as an iterative process by which a system builds increasingly more complex structures, and uses these structures to behave in interaction with the environment. However, most of them fall under Brook's critics. Moreover, most of the methods only

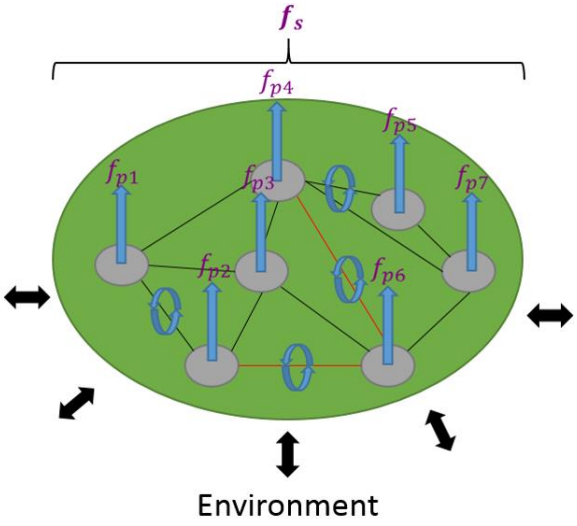

Figure 3: A schematic view of an AMAS system. The functionality f_s provided by the system is more than the sum of each agent functionality f_{p_i}. It is the result of interactions between agents and the environment.

took interest on knowledge creation, avoiding the problem of knowledge exploitation.

We agree with Brooks and Pfeifer vision of intelligence. That means that to be truly adaptive, the design of an *XS Function* should not fall into Brook's critic of abstraction. An XS *Function* must then exploit all source of information as a signal without making any abstraction on it. Semantic is then prohibited. On contrary, each signal has to be considered the same way, as a raw observation of the world.

Furthermore, as we cannot make a separation between the body and the mind, the learning process allowing self-adaptation has to be self-aware of its own activity and its consequences on what it senses from its environment. Learning from the consequences of my own embodiment relation (which means consequence of my own activity) will allow the system to sense any changes on this relation, either this changes come from a modification of system's body or environment. The learning process should be made through interaction with the physical and social environment by which a complex behavior emerges.

To be usable by any kind of user, the adaptation process must not require any expertise. *Learning from Demonstration* (Argall, 2009) appears then to be a promising approach. Learning from Demonstration is a paradigm to dynamically learn new behaviors from demonstrations performed by a human. The process of demonstration does not require expertise from the user on the controlled system while allowing the system to capture the user's needs.

On the next section, we present our contribution to enable the *XS Robotic* vision. This contribution is a combination of the *Adaptive Multi-Agent System approach* and *Learning from Demonstration*.

5 ADAPTIVE LEARNER BY EXPERIMENTS

Through the scope of XS Robotic, the problem of integration of robotic components is a problem of self-adaptation. We then need to propose an algorithm that enable each device to self-adapt. On this section, we present our contribution, ALEX, an adaptive multi-agent system designed to learn from demonstration performed by a tutor. Its design is based on the Adaptive Multi-Agent System (AMAS) approach.

5.1 AMAS approach

The *Adaptive Multi-Agent System* approach (Gleizes, 2012) addresses the problematic of complex systems with a bottom-up approach where the concept of cooperation is the core of self-organization. The theorem of functional adequacy (Camps, 1998) states that:

> "For all functionally adequate systems, there is at least one system with a cooperative internal state that realizes the same function in the same environment"

A general definition of *cooperation* could be the golden mean between altruism and selfishness (Picard, 2005). The role of an AMAS designer is to identify non cooperative situations and to propose mechanisms to anticipate or resolve such situations. The agent detecting a non-cooperative situation automatically triggers those mechanisms. Three mechanisms allow repairing or anticipating a non-cooperative situation (Capera, 2003):

- **Tuning**: the agent adjusts its internal state to modify its behavior,
- **Reorganization**: the agent modifies the way it interacts with its neighborhood,
- **Evolution**: the agent can create other agents or self-suppress when there is no other agent to produce a functionality or when a functionality is useless.

The system will then self-organize to stay in a cooperative state. From cooperative interactions between the system's entities emerges a global function that is more than the sum of the parts (Figure 3).

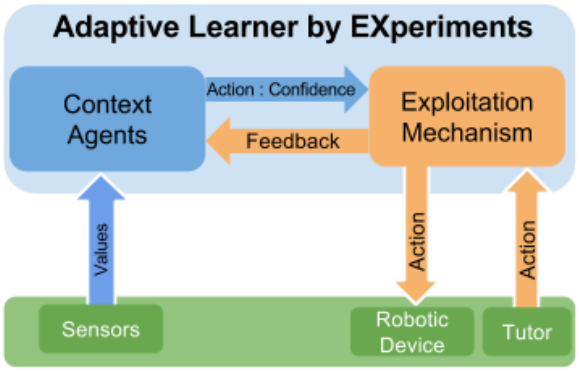

Figure 4: ALEX architecture

The approach proposes a methodology called ADELFE that guides the designer of an AMAS system (Bonjean, 2014).

5.2 Learning from Demonstrations

Learning from Demonstration, also named *Imitation Learning* or *Programming by Demonstration*, is a paradigm mainly studied in the robotic field that allows systems to self-discover new behaviors (Argall, 2009). It takes inspiration from the natural tendency of some animal species and humans to learn from the imitation of their congeners. The main idea is that an appropriate controller for a robotic device can be learnt from the observation of the performance of another entity (virtual or human) named as the ***tutor***. The tutor can interact with the system to explicit the desired behavior through the natural process of demonstration. A demonstration is then a set of successive actions performed by the tutor in a particular context. The learning system has to produce a mapping function correlating observations of the environment and tutor's actions to its own actions. The main advantage of such technique is that it needs no explicit programming or knowledge on the system. It only observes tutor's actions and current system context to learn a control policy and can be used by end-users without technical skills.

The paradigm has been used on a wide range of applications such as autonomous car following (Lefèvre, 2015), robot trajectory learning (Vukovic, 2015) or robot navigation in complex unstructured terrain (Silver, 2010). Recent surveys (Billard, 2008) (Argall, 2009) propose an overview of the LfD field illustrating a wide variety of applications. Our interest is not to focus on one particular application. On the contrary, we want to deal with any kind of ambient robotic system.

5.3 ALEX architecture and general behavior

In accordance with the ADELFE (Bonjean, 2014) methodology, we designed ALEX (Adaptive Learner by Experiment), an Adaptive Multi-Agent System, to learn to control a system from demonstrations.

On the rest of this section, we present ALEX architecture and focuses on Context agents, which are the core of the learning process.

5.3.1 ALEX architecture

An ALEX instance is designed to control a robotic device (an effector) by sending actions to it. Those actions are changes of the current state of the robotic device. An ALEX instance is in constant interaction with its environment from which it receives actions from its tutor and a set of sensors values. ALEX observes the current state of all accessible sensors, the action performed by the tutor and in response sends the action to be applied by the controlled robotic device. ALEX is composed of two components, an Exploitation mechanism and a set of Context agents. The figure 4 illustrates ALEX architecture.
The Exploitation mechanism is responsible for sending actions to the robotic device. In order to do so, it receives both the action performed by the tutor and a proposition of action from the set of Context agents. By comparing the action realized by the tutor to the proposition made by Context agents, the Exploitation mechanism can generate a feedback that is sent to the set of Context agents.
Context agents are the core of the learning. They are responsible of making action proposition based on what they have observed of previous tutor actions. More details about this architecture can be found on previous work (Boes, 2015).
On the rest of this section, we present the behavior of Context agents.

5.3.2 *Context-agents* behavior

The term context in this paper refers to all information external to the activity of an entity that affects its activity. This set of information describes the environment as the entity sees it (Guivarch, 2014). ALEX interacts with a tutor (virtual or human) which performs a set of demonstration. A demonstration consists in the performance of an action under a particular context. Each time an action is performed, ALEX correlates the effect of the performance of this action on the current situation to effects of this action on the environment. ALEX receives a set of signals O from the environment that describes the current situation. Each signal $o_n \in O$ is a continuous value associated to a unique identifier. The identifier is used to discriminate signals and has no semantic value.

ALEX is composed of a set of *Context Agents*. A *Context agent* is a tripartite structure composed of a context description, an action, and an expectation of the utility of the action under this particular context:

$$< context, action, utility >$$

At start, the set of *Context agents* is empty as ALEX possesses no a priori knowledge. *Context agents* are autonomously and dynamically created. *Context agents* receive signals from the environment (from sensors) which they use to characterize the current context. To build its context description, a Context agent associates to each signal O from the observation space a set of two bounds $< o_{min}, o_{max} >$. Every time the observation space O is included to its context description, a Context agent makes an action proposal. This proposal could be interpreted as "if you do this particular action under this particular context, you can expect this particular utility". At its creation, a context agent is associated to a unique action. The role of a context agent is then to both learn the context description and the utility associated to its action thanks to feedbacks it perceives from its tutor and signals it perceives from the environment. When the tutor is not acting on the system, *Context agents* are responsible of system autonomy. Then they must cooperate to perform an effective control on the device that satisfies the tutor.

Context agents dynamically manage their context description by either reducing or expanding their bounds (figure 5) in interaction with the tutor. Each time the tutor performs an action, *Context agents* observe the tutor activity and compare it to their own action.

Four adaptation processes can occur:

- **Expansion**: When the tutor performs an action that is close to a Context agent's context, and this Context agent proposes the same action, the Context agent can manage its bound to integrate the current situation.
- **Reduction**: When the tutor performs an action that is different to the action proposed by a Context agent, but this Context agent context description include this situation, the Context agent updates its bound to exclude the current situation.
- **Suppression**: By adjusting its bounds, a Context agent can find himself in a situation where $o_{max} < o_{min}$. Such situation produces the destruction of the Context agent.
- **Creation**: When no Context agent proposes the user action (and no Context agent can expand to represent the situation), the system autonomously create a new Context agent to represent the tutor's action.

Bounds are managed by *Adaptive Value Trackers*, a software component that is able to find the value of a dynamic variable in a given space through successive feedbacks. More information on Adaptive Value Trackers can be found on previous work (Guivarch, 2015). The main advantage of this context description is that no semantic on signals is used: it only observes variation of signal values. In parallel to the adaptation of bounds, *Context agents* maintain a utility value. This utility value helps *Context agents* to disambiguate situations where many *Context agents* with different actions propose to perform an action. Utility is then used to determine which the best action to perform is. Utility value is based on Context agent history. The more a Context agent can makes action proposal that is different to the user activity, the more its utility value is low. On contrary, the more the Context agent proposition has been confirmed by user activity, the more the agent is confident in its utility. This value is managed by the function:

$$C_{t+1} = C_t * (1 - 0.8) + F_t * 0.8$$

where C_t is the utility value at time t, and F is the similarity with the tutor where 1 means that the context is proposing the same action than the tutor and 0 means that two actions are different.

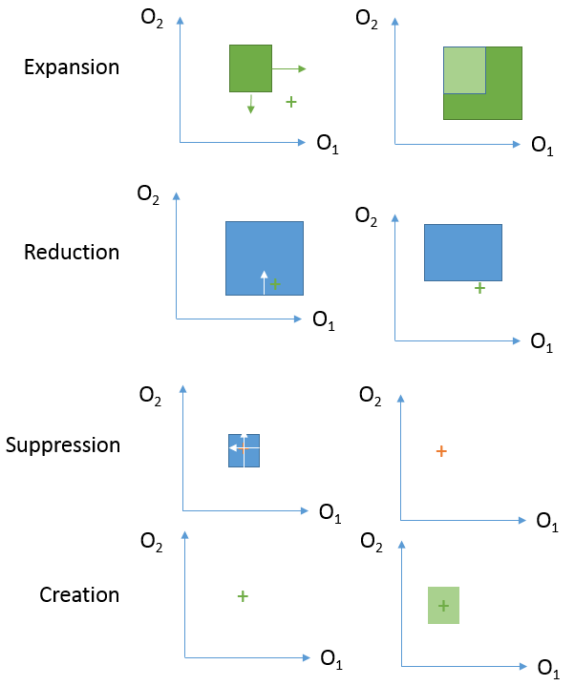

Figure 5: Context agent bounds management example. The cross represents the current situation and its color the action performed by the user.

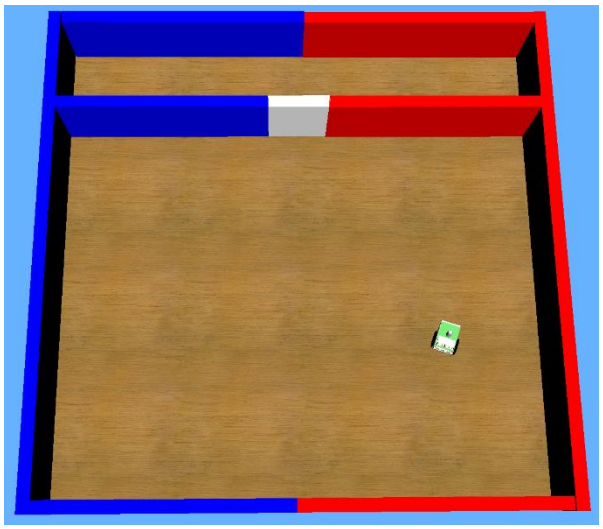

Figure 6: A view of the simulation. The rover evolves in an arena and has to reach the white door.

Whenever the tutor performs an action on the device, *Context agents* use this action to self-adapt by adjusting their bounds and their utility value. However, when the tutor is not acting, *Context agents* use the acquired knowledge to cooperate and control the device.

5.4 ALEX previous work

ALEX has been previously evaluated on a collection task (Verstaevel, 2015). The experiment illustrates the advantages of our approach for end-user, allowing a natural way to express their needs. A tutor controlling a two-wheeled robot demonstrates a collecting task. By comparing the number of artefact collected by the tutor in 5 minutes to the score performed by the rover in autonomy, the results show that ALEX managed to learn to perform the activity more efficiently than the tutor does. The *Context learning* architecture has been abstracted and applied to various domains (Boes, 2014). On this article, we change of viewpoint and want to illustrate advantages of our approach for designers. The next section proposes a use-case study to show those advantages.

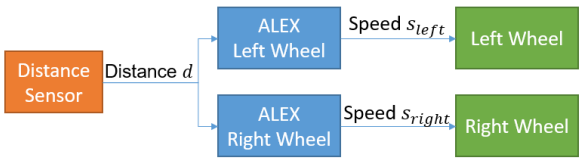

Figure 7: Architecture of the first experiment. Each ALEX receives the distance value and has to associate to this value the adequate speed.

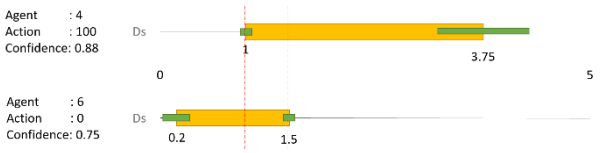

Figure 8: A comparison of two Context agents 4 and 6 extracted from the first experiment. The two Context agents propose a different action under the same context leading to ambiguity.

6 A USE-CASE STUDY

We propose to study the benefits of our approach for the design of the following application:

> A two wheeled rover has to go from an area A to an area B. The passage between the two areas is only possible through a door. The area A may be populated with obstacles. The rover then has to navigate through the area to reach the gate, and then go through it. The experiment is complete when the rover is in the area B and the door is closed.

The experiment is implemented on Webots®, a robotic simulator. The arena is composed of two areas, separated with a white door. The walls on the left part of the arena are blue, the walls on the right part of the arena are red (see figure 6). The ALEX implementation we used is developed in Java and is the same for all experiments. At each time step, an ALEX instance receives data values, the action performed by the tutor and in response, provides the action to be performed.

Each experiment is decomposed in two phases. First, the designer performs a complete demonstration of the activity during which he takes control of the rover. This first phase allows each ALEX instance to acquire Context agents. Secondly, the rover performs the task autonomously by using the previously learnt Context agents. The designer then observes the activity to determine if or not the rover behavior is satisfying.

In case of a failure in the reproduction of the task, the designer can extract and analyze Context agents' structure.

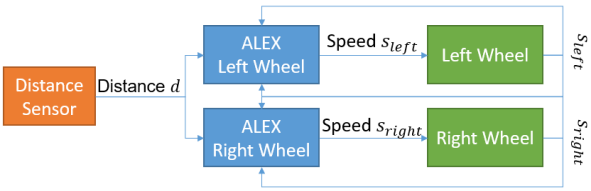

Figure 9: Modified architecture of the first experiment. Each ALEX now receives the distance and the current speed of both wheel.

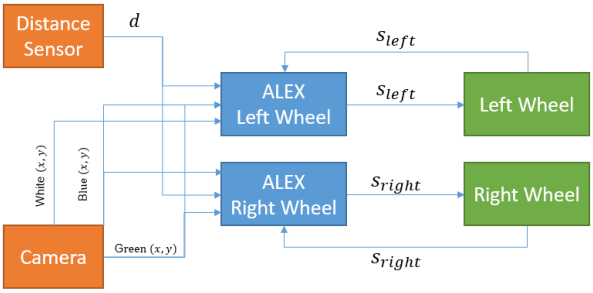

Figure 10: The second experiment architecture. A camera is added providing three new couple of value (x,y) for each detected color.

6.1 First experiment: avoiding obstacles

In accordance with the XS Robotic vision, the designer first identifies the functionalities involved in the application.

> *The rover is composed of two wheels, each wheel controlling its own speed value from* -100 *to* 100*. Each wheel is then an XS function of action.*

Then the designer needs to provide to the rover some perception about its surrounding environment.

> *As the task involves avoiding obstacles, a distance sensor is identified as required to navigate through the area. The distance sensor is then an XS function of perception.*

Once both functions of action and perception have been identified, the designer can realize its first experiment and try to teach to the rover the task. The figure 7 illustrates the architecture of this first experiment.

After this first demonstration, we observe that the rover fails in its navigation task. By observing the Context agents inside each ALEX instances, we found that using only distance value leads to ambiguities in the demonstration. The phenomenon is observable in figure 8. The figure shows the structure of two Context agents after the demonstration. The yellow area corresponds to the values of the distance sensor where the Context agent is valid. The green area to the values where the Context agent is extensible. Those two Context agents are extracted from the right wheel ALEX. They propose a different action. The first one propose to go at a speed of 100 whereas the second one proposes to go at a speed of 0. If we observe the two validity range of the Context agents, we observe an overlap. This overlap means that the two Context agents will propose their action in similar situation, leading to ambiguity. The reason is that using only an ultrasound sensor is not well enough to discriminate each situation. When the rover is at a

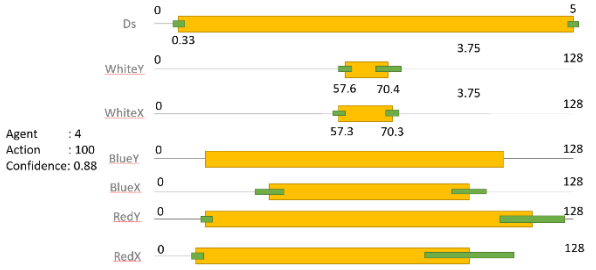

Figure 11: A particular Context involved in the last experiment. This Context agent is a lot more sensitive to the White (x,y) value than the other.

particular distance, it can express either that the rover is approaching an obstacle or moving away.

> *To disambiguate those situations, the designer propose to use the current speed value of both wheels as an input to the ALEX instance.*

A new demonstration is performed with this new architecture (figure 9) and the rover now succeed to navigate through the area. However, as it cannot differentiate a wall from the door, the rover fails to learn to reach the door.

6.2 Second experiment: reaching the door

> *As the rover needs to differentiate walls and the door, the designer proposes to add a Camera on the rover to recognize characteristics of the objects to be detected. Using a detection algorithm, the camera can provide visual information about the environment of the rover. As walls and the door have different colors, the camera identifies the coordinate* $(\boldsymbol{x}, \boldsymbol{y})$ *of the center of each of the three color blue, white and green.*

A Camera is added to the simulation to provide new data to the rover. Each ALEX instance now receives, in complement of the previous data, the coordinate $(\boldsymbol{x}, \boldsymbol{y})$ of the center of each color (see figure 10). If no artefact of one color is detected,

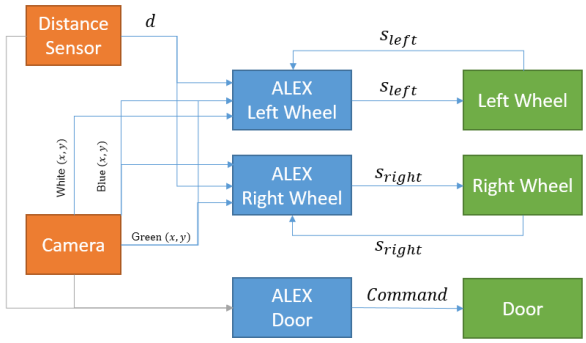

Figure 12: The architecture of the last experiment. The door is now controlled by and ALEX instance and receives the same information than the other ALEX.

the coordinates provided are (-1,-1). The addition of the camera does not involve any modification on the ALEX instances. The tutor then performs another demonstration of the task.

Now the rover manages to navigate inside the arena and reach the door. By observing the structure of the *Context agents* involved in this experiment, we found that the agents involved in the part of the activity reaching the door have learnt to be less sensible to the blue and red coordinates. One example of those agents is visible in figure 11. The validity range associated the signals $\boldsymbol{WhiteX}$ and $\boldsymbol{WhiteY}$ are smaller than the one associated to $\boldsymbol{BlueX}$ and $\boldsymbol{BlueY}$, and $\boldsymbol{GreenX}$ and $\boldsymbol{GreenY}$. As the activity only involves identifying the white door, the other data are unrelated. Our designer can exploit this information to remove unused data from the system. However, the rover failed to complete the task. While it managed to reach the door, the rover failed to open it as its engines were not powerful enough.

6.3 Last experiment: opening the door

For the third experiment, the door is equipped with a motor. An ALEX instance is associated to the door and must learn when it has to be open and when it has to be closed. For thus, the door receives the same data than the two ALEX instances controlling the wheels (Figure 12). Adding this new effector does not involve any action on the pre-existing devices and previously learnt Context agents can be kept.

The tutor performs a final demonstration of the task, demonstrating to each component the desired behavior. At last, the rover managed to learn to reach the area B. The door correlated the action of opening to a low value of the distance sensor signal and the coordinate $WhiteX$ and $WhiteY$ near the center of the screen. The rover and the door managed to collaborate without direct communication. As they share perception, they have enough information to coordinate their activity.

6.4 Synthesis

The whole experiment illustrates both the Extreme Sensitive Robotic paradigm and ALEX capacity to learn in interaction with human. The design of a robotic application with ALEX allows the designer to focus on the desired functionality and to let to ALEX the duty to find correlations between the performing of an action and the state of sensors. An application can be designed incrementally by gradually adding new sensor and effectors. The usage of a learning technic based on self-observation allows the designer to point out situations of ambiguity or situations where some data are useless. Adding a new sensor or a new effector is not a complex task anymore as each component can self-adapt.

7 CONCLUSION

The design of a controller for a robotic application is a complex task. In this article, we propose to give to each device the ability to self-adapt in interaction with its environment. We propose an approach named "*Extreme Sensitive Robotic*" which focuses on the bottom-up design of robotic application. To enable each device to self-adapt, we proposes the use of ALEX, an adaptive multi-agent system based on Context-Learning.

In previous work, we have shown the advantages of our approach for end-user, enabling the robotic device to automatically learn a behavior from demonstrations. In this article, we have taken the viewpoint of the designer of such a system. Through a use case, we have shown that the combination of Extreme Sensitive Robotic and ALEX could help designers to incrementally build their system. By studying the *Context agents* dynamically created by ALEX, the designer can point out ambiguity in signals and decide to increase the perception capacity of the system. The same analysis can lead the designer to add new effectors on the system. As each effector is designed to be self-adaptive, the appearance or disappearance of an effector or of a new sensor does not imply to re-act on the previously deployed effector.

However, the study of *Context agent* is in this paper still hand-performed by the designer. Then, work is being made to allow *Context agents* to automatically discover situations of ambiguity where data are missing and to automatically solve those situations. Such automatic process involves to distribute intelligence inside each sensor which will become a cooperative agent by locally analyzing how they interact with *Context agents*.

REFERENCES

B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, *Robotics and autonomous systems* 57 (5) (2009) 469–483

A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot programming by demonstration, in: Springer handbook of robotics, Springer, 2008, pp. 1371–1394.

J. Boes, J. Nigon, N. Verstaevel, M.P Gleizes & F. Migeon. The Self-Adaptive Context Learning

Pattern: Overview and Proposal. In: Proceedings of the Ninth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT2015), 2015

N. Bonjean, W. Mefteh, M.-P. Gleizes, C. Maurel, F. Migeon, Adelfe 2.0, in: *Handbook on Agent-Oriented Design Processes*, Springer, 2014, pp. 19–63.

R.A Brooks. Elephants don't play chess. Robotics and autonomous systems, 6(1):3-15, 1990.

D. Capera, J.-P. Georgé, M.-P. Gleizes, P. Glize, The amas theory for complex problem solving based on self-organizing cooperative agents, in: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003, IEEE, 2003, pp. 383–388.

Frank Guerin. Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review*, 26(02):209-236, 2011.

M.-P. Gleizes, Self-adaptive complex systems, in: Multi-Agent Systems, Springer, 2012, pp. 114–128.

Guivarch, V., Camps, V., & Péninou, A. Context awareness in ambient systems by an adaptive multi-agent approach. In *Ambient intelligence*, Springer Berlin Heidelberg, 129-144, 2015

S. Lefèvre, A. Carvalho, F. Borrelli, Autonomous car following: A learning-based approach, in: *Intelligent Vehicles Symposium* (IV), 2015 IEEE, IEEE, 2015, pp. 920–926.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials*, IEEE, 16(1), 414-454, 2014.

Picard G and Glize P, Model and Experiments of Local Decision Based on Cooperative Self-Organization. In: *Second International Indian Conference on Artificial Intelligence* (IICAI'05), 2005.

Rolf Pfeifer and Josh Bongard. How the body shapes the way we think: a new view of intelligence. *MIT press*, 2006.

D. Silver, J. A. Bagnell, A. Stentz, Learning from demonstration for autonomous navigation in complex unstructured terrain, *The International Journal of Robotics Research*, 2010.

Alan M Turing. Computing machinery and intelligence. Mind, pages 433-460, 1950.

N. Verstaevel, C. Régis, M.P Gleizes, F. Robert. Principles and Experimentations of Self-organizing Embedded Agents Allowing Learning from Demonstration in Ambient Robotic. *Procedia Computer Science*, vol. 52, p. 194-201, 2015.

N. Vukovi´c, M. Miti´c, Z. Miljkovi´c. Trajectory learning and reproduction for differential drive mobile robots based on gmm/hmm and dynamic time warping using learning from demonstration framework, *Engineering Applications of Artificial Intelligence 45*, 388–404, 2015.

Weiser, M., The computer for the 21st century. *Scientific american*, 265(3):94–104, 19

J. Zlatev and C. Balkenius. Introduction: Why epigenetic robotics? *Epigenetic Robotics*, 2001.