University of Wollongong
Research Online

SMART Infrastructure Facility - Papers

Faculty of Engineering and Information Sciences

2015

Principles and Experimentations of Self-organizing Embedded Agents Allowing Learning from Demonstration in Ambient Robotic

Nicolas R. Verstaevel
University of Wollongong, nicolasv@uow.edu.au

Christine Regis
Universite Paul Sabatier

Marie-Pierre Gleizes
University of Toulouse

Fabrice Robert
Sogeti High Tech

Follow this and additional works at: https://ro.uow.edu.au/smartpapers

 Part of the Engineering Commons, and the Physical Sciences and Mathematics Commons

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Principles and Experimentations of Self-organizing Embedded Agents Allowing Learning from Demonstration in Ambient Robotic

Abstract

Ambient systems are populated by many heterogeneous devices to provide adequate services to its users. The adaptation of an ambient system to the specific needs of its users is a challenging task. Because human-system interaction has to be as natural as possible, we propose an approach based on Learning from Demonstration (LfD). However, using LfD in ambient systems needs adaptivity of the learning technique. We present ALEX, a multi-agent system able to dynamically learn and reuse contexts from demonstrations performed by a tutor. Results of experiments performed on both a real and a virtual robot show interesting properties of our technology for ambient applications.

Keywords

allowing, learning, robotic, principles, experimentations, demonstration, self-organizing, ambient, embedded, agents

Disciplines

Engineering | Physical Sciences and Mathematics

Publication Details

Verstaevel, N., Regis, C., Gleizes, M. & Robert, F. (2015). Principles and Experimentations of Self-organizing Embedded Agents Allowing Learning from Demonstration in Ambient Robotic. Procedia Computer Science, 52 194-201.

This journal article is available at Research Online: https://ro.uow.edu.au/smartpapers/241

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 194 – 201

Procedia Computer Science

The 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015)

Principles and experimentations of self-organizing embedded agents allowing learning from demonstration in ambient robotic

Nicolas Verstaevel[a,b,*], Christine Régis [a], Marie-Pierre Gleizes[a], Fabrice Robert[b]

[a]IRIT, Université Paul Sabatier, 118 rte de Narbonne, 31062 Toulouse Cedex
[b]Sogeti High Tech, 3 Chemin Laporte, 31300 Toulouse

Abstract

Ambient systems are populated by many heterogeneous devices to provide adequate services to its users. The adaptation of an ambient system to the specific needs of its users is a challenging task. Because human-system interaction has to be as natural as possible, we propose an approach based on Learning from Demonstration (LfD). However, using LfD in ambient systems needs adaptivity of the learning technique. We present ALEX, a multi-agent system able to dynamically learn and reuse contexts from demonstrations performed by a tutor. Results of experiments performed on both a real and a virtual robot show interesting properties of our technology for ambient applications.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Conference Program Chairs

Keywords: Adaptive agents, Cooperative multi-agents systems, Robotic in ambients systems, Machine Learning, Context-Awareness

1. Introduction

Once confined in a science of control in structured environments, researches on robotic are now considering the integration of intelligent systems in the real world for applications where tasks are multiple, complex and evolutive[1]. Service robotic differs from its industrial version by the interest in providing services to humans. Consequently, researchers now consider the use of robotic components in ambient applications[2].

Ambient systems are characterized by their high dynamic and their complexity. Many heterogeneous devices can appear and disappear along the system lifecycle and interact opportunistically together. According to the definition of Russell and Norvig, the environments of such systems are[3]:

- **Inaccessible**: each device composing the system has a partial observation of the environment.
- **Continuous**: considering applications in the real world, the number of observations and actions is not discrete.
- **Non-deterministic**: consequences of performed actions in the real world could not be determined in advance with certainty.
- **Dynamic**: system's actions, user activity, appearance and disappearance of devices may change the environment.

Consequently, designing an *ad hoc* controller of an ambient system is a complex task that requires lot of knowledge. This complexity is increased if we take into account that users have multiple, specific and often changing needs. Bringing the ability to learn and adapt to users' needs is then a particularly challenging task[4]. To be as natural as possible, such learning ability needs

* Corresponding author. Tel.: +33-5-61-55-62-36.
E-mail address: verstaev@irit.fr

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.056

to rest on a process that does not require any kind of knowledge for users (i). Furthermore, it needs both **genericity**, to be applicable on any kind of devices with any kind of users, and **openness** properties to deal with the appearance and disappearance of devices. The genericity and openness properties require then using *agnostic learning* technics that makes as few assumptions as possible[5] (ii). To deal with (i) and (ii), we propose to use Learning from Demonstration, a paradigm to dynamically learn new behaviours, as the engine of self-organization in ambient applications.

The paper is organized as follows: first, we present the problematic and challenges of LfD in ambient systems. Then, we present ALEX our solution to handle this challenge. An experiment illustrating ALEX behaviour on a collecting task is then provided. At last, conclusion will discuss perspectives and future works.

2. Learning from demonstration

2.1. General principle

Learning from Demonstration, also named "imitation learning", is a paradigm mainly studied in the robotic field that allows systems to self-discover new behaviours[6]. The main idea is that an appropriate controller can be learnt from the observation of the performance of another entity (virtual or human) named as the *tutor*. A tutor can interact with the system to explicit the desired behavior. This interaction is done through the natural process of *demonstrations*, which consists in the performance of an action in a particular context. The system then learns a mapping function correlating observations of the environment to tutor's action. The main advantage of such technique is that it needs no explicit programming or knowledge on the system. It only observes tutor's action and current system context to learn a control policy. Recent surveys[6,7] proposed an overview of the LfD field. Billing and Hellström[8] propose a complete LfD formalisation.

LfD is a problem of imitation as an entity tries to produce a behaviour similar to another entity. A tutor evolving in a world Ω can perform a set of actions A (A could be empty). The *tutor* follows a policy (1) that can associate to any world state a particular action. It is supposed that (1) is the optimal policy to satisfy the user.

$$\Pi: \Omega \rightarrow A \tag{1}$$

An imitator (named as the *learner*) disposes of a set of observations O (named *observation space*) on the states space Ω (named the *world*) and follows another policy (2) in order to produce a behaviour similar to the observed one.

$$\Pi': O \rightarrow A \text{ such as } \Pi' \equiv \Pi. \tag{2}$$

In many cases, the tutor and the system have a different observation of the world. It is particularly true in real world problems with human tutors where the world is observed by the system through sensors whereas the human observes it through its own body. It results in a problem of *perceptual equivalence*[9]. The *tutor* demonstrating a particular behavior can observe modifications of the world that the *learner* cannot perceive. However, equivalences of perception can be found by the system. For example, a user is cold and turns the heating on. Observing through sensors that a user is cold is complex, but observing the temperature, wind and humidity levels is easily feasible. A learner can make correlation between the current situation and the action of turning the heating on and learns that it is necessary to turn the heating on.

The learner has to find correlations between its own observations and the performance of an action by the tutor. This raises the question of the possible lack of equivalence. A tutor can perform a demonstration dependent of a phenomenon that the learner cannot observe. In this paper, we consider that the learner disposes of sufficient observations to perform the task. Nevertheless, some clues to handle this problem are proposed on section 5 as perspectives.

2.2. Lfd and ambient systems

Ambient systems are rich of interaction possibilities for users. We propose to exploit the inherent interactivity of ambient systems in order to learn and adapt from users activity. LfD then appears to be a good paradigm to guide system adaptation. Therefore, users can be seen as system's tutors. Each user's action on a device is then seen as a demonstration of the desired behaviour. The idea is that if a user has to act on a particular device, the reason is that the service provided by this device is not satisfying anymore. The device can then use this information to self-adapt. However, ambient systems have some particular properties that are challenging for LfD.

An ambient system is **open**, which means that entities can appear and disappear during system activity. This openness property is challenging for LfD, as it does not allow doing assumptions on system's composition. With respect to the openness property, it must be considered that the set of observations O is a priori unknown. Moreover, to provide a **generic** learning technic, it has also to be considered that the set of actions A is a priori unknown. Then, a learner on an ambient system must adapts its policy to integrate new observations and actions. This adaptation must be dynamic and transparent to users to provide a good quality of service. Thus, approaches separating learning phase from exploitation (supervised learning) appear to be not pertinent.

Users are also a source of complexity as they have specific and often changing **needs**. In ambient systems, many different

users have day-to-day interactions with the system. Capturing users' needs in such a system cannot be an *ad hoc* process and has to be made along system lifecycle. As users can often change their needs and contradict themselves. The system dynamically adapts its policy to integrate any change in the user's need. An approach based on user profiles appears to be not relevant.

LfD in ambient systems needs for ***adaptivity*** to the task to perform (*what* to do), to the reasons to do it (*when* to do) and to the users (*whom* to imitate).

2.3. An extremely sensitive system

Nevertheless, *what* and *when* are not the only questions to bring adaptivity in ambient systems. The classical approach considers the system as a whole, which means the system is seen as an omniscient process that exercises a direct control over all its parts. However, this centralized approach shows some limitation for highly dynamic and open systems where the observability is local. Works among the community now consider decentralized approach where there is no supervisor of the activity[1,4,11]. This shift of paradigm raises the question of *who* learns. In literature concerning robotic, the answer is often "the robot", as the entity composed of sensors and effectors. The more there are sensors and effectors, the more complex the robot is. The same reasoning can be applied to ambient systems. We argue that this vision is restrictive and propose what we call "Extreme Sensitive Robotic"[10].

The eXtreme Sensitive Robotic (XS Robotic) vision considers each functionality as an autonomous entity. A robot is then composed of many eXtreme Sensitive Functions (XS Functions), globally one for each sensor and effector. An XS Function is sensitive (such as a temperature or ultrasound sensor) or effective (a motor, a led). Each XS Functionality has to be designed as *nonfinal*, which means that the functionality can self-evolve. Self-observation capacities are the key for sensing variation in the environment and adapt in response. With this approach, the production of a complex behavior by a robot emerges from the interaction between the different XS Functionalities and the environment. There is thus no difference of design between a single robot, a multi-robot application or a complex ambient system. All those systems are composed of independent functionalities that need to cooperate to perform a global function that is greater than the sum of its parts.

The next section presents ALEX, a multi-agent system for learning by demonstration in ambient robotic, built upon the eXtreme Sensitive Robotic vision. Its conception is based on the Adaptive Multi-Agent System (AMAS) approach and uses recent results on context-aware learning[11].

3. ALEX: Adaptive Learner by EXperiments

The Adaptive Learner by Experiments (ALEX) is a multi-agent system designed to learn from demonstration to control a functionality (as described in section 2.2). It has been built upon the Adaptive Multi-Agent System approach (AMAS)[12], which addresses the problematic of complex systems with a bottom-up approach where the concept of cooperation acts as the core of self-organization. The theorem of functional adequacy[13] states: "for all functionally adequate systems, there is at least one system with an internal cooperative state that realizes the same function in the same environment". The role of an AMAS is then to automatically detect and repair non-cooperative situations by self-organizing to reach a functionally adequate state. ALEX has been developed according to the ADELFE[14] methodology that guides the design of an adaptive system.

3.1. Context-Learning of an XS functionality

Section 2 illustrates that making an exhaustive list of all situations that an ambient system may be faced to is impossible. It is then necessary for the system to dynamically adapt to contexts. The term *context* refers in this paper to all information external to the activity of an entity that affects its activity. This set of information describes the environment[11]. A system capable of exploiting context information is called "context-aware".

ALEX is designed to continually interact with its environment and dynamically learn all the different contexts that can occur. It associates to each context the adequate action to perform. ALEX uses self-observation capacities to dynamically build correlations between the performance of an action and effects of this action on the environment. For thus, an ALEX is composed of a non-finite set of *context agents* that makes a collective control over an XS functionality. The set of context agents is empty and ALEX dynamically and autonomously creates those context agents. In this section, we describe the structure and behaviors of context agents. The next section will provide an experiment to illustrate how contexts make an efficient control of XS functionalities on a concrete application. Context agents are described by decomposing their behaviors in two kinds. The *nominal* behavior is the normal behavior that the agent performs when the system is in a functionally adequate state. The *cooperative* behavior is a subsumption of the nominal behavior that occurs when the agent needs to self-organize to bring the system toward a functionally adequate state.

3.2. Context agents

3.2.1. Context agent nominal behavior

A *Context Agent* associates a low-level context description V (see section 3.2.2) to a unique action. An action corresponds to maintaining or adjusting the XS functionality. This action is given at the agent creation and never changes (see section 3.2.3.2). It can be a high-level command (such as "go Forward", "go Left") or a low-level command ("speed at 42%", "x=2.1",) depending on the functionality to control. A Context Agent receives signals from the environment and uses them to characterize current context. When the observed environment O corresponds to its low-level context description V, it decides to perform its associated action. Otherwise, it does nothing. When a context agent performs an action, the agent is said *selected*.

3.2.2. Low-level context description

Value ranges (named *validity ranges*) manage the low-level context description. A *Context Agent* receives a set of observations O from the environment. Each $o \in O$ is a continuous value such as $o \in [o_{min}, o_{max}]$. A validity range $v \in V$ is associated to each observation such as v_o corresponds to the validity range associated to the observation o. The set of all validity ranges V is named the *validity domain*.

A validity range is composed of two values, v_{min}, v_{max} such as $[v_{min}, v_{max}] \subseteq [o_{min}, o_{max}]$. A validity range is said *valid* if and only if as $o \in [v_{min}, v_{max}]$. Then, a context agent is said *valid* if and only if $\forall v \in V, v$ is valid. Validity ranges allow the context agent to determine if O is similar to V.

Adaptive Value Range Trackers (AVRT) manage validity ranges. AVRT is an extension of Adaptive Value Tracker (AVT)[15], a tool that can dynamically find a value from feedbacks greater, lower and good.

3.2.3. Context agent cooperative behavior

At each time step, only one action can be performed over the controlled functionality. Thus, each context agent has to determine if its action is the most adequate. To be in a functionally adequate state, the system then needs that only one context agent proposes an action. However, situations where more than one context agent proposes an action can occur. Those situations are non-cooperative situations (NCS). Three NCS can occur. The first one (a) occurs when two (or more) agents propose to perform an action (regardless of the nature of the action). The second one (b) occurs if the context agent proposes an action that is not in adequation with the tutor's one. The last one (c) occurs when no context agent proposes an action. To solve these situations, context agents follow a cooperative behavior to dynamically self-organize.

3.2.3.1. Confidence

A *Context Agent* determines a *confidence* value that represents the relevance of the action to perform. This confidence value allows context agents to compare their actions. If a context agent wants to perform an action with a confidence value lower than another agent, it has to self-organize its context description to exclude the current situation. Thus, the next time the same context occurs, the context agent will not try performing its action.

The confidence value v is ruled by a lambda function (3).

$$v_{t+1} = v_t \times (1 - \alpha) + F_t \times \alpha \quad (3)$$

v_{t+1} is the confidence value at the step $t + 1$. $F_t \in 0|1$ is a feedback value and $\alpha \in [0,1]$ is a given parameter that models the importance of the feedback. Each time a context agent makes a correct proposal, $F_t = 1$. Otherwise, $F_t = 0$. In our model, α is fixed at *0.8*. The more the context agent makes correct proposals, the more its confidence value increase. At creation, $v = 0.5$.

The value is then used in the decision process of the context agent. Each context agent can observe the current best confidence value to decide if its action is the most adequate. Confidence allows solving the ambiguity made by the non-cooperative situation (a).

A low confidence means that the context agent is often mistaken. A context agent with a low confidence level will decide to destroy himself to not disturb the system. The minimal confidence value is set at *0.1*, meaning that any context agent with a confidence lower than *0.1* will self-destroy.

3.2.3.2. Adequation with the tutor's action

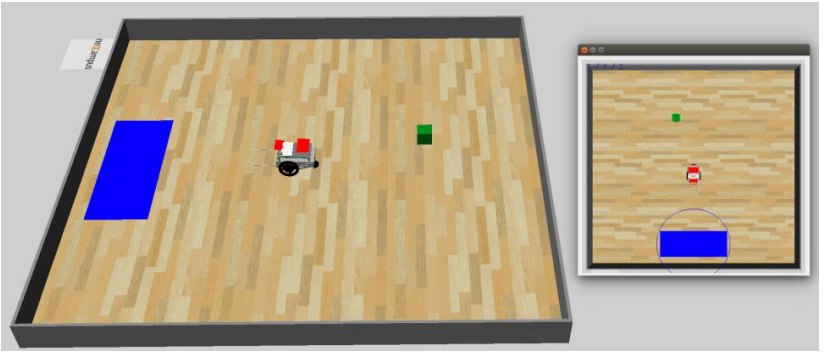

Figure 1 : The experiment in Webots™. On the left, the rover inside the arena. On the right, the camera detection.

A tutor can interact at any time with the system by demonstrating the desired behavior. The demonstration consists in the performance of a particular action $a \in A$. Thus, at any time step, there is a value $a_{tutor} \in A$ corresponding to the tutor's action at this time step. Each context agent can perceive this value. If $a_{tutor} = \emptyset$, the tutor has performed no action. Thus, it is context agent's duty to find the most adequate action to perform. However, if $a_{tutor} \neq \emptyset$, context agents can observe tutor's action. If a context agent finds himself *valid* but proposes a different action than the tutor's one, it has to self-organize to exclude the current situation. Each *valid* validity range manages its bounds to exclude the current situation. If the set of observations O contains values that are not associated to a validity range (for example, when a new sensor is added on the system), the context agent adds a new validity range to its validity domain corresponding to the new observation. However, if a context agent proposes an adequate action, it reinforces its. This allows solving the ambiguity made by the non-cooperative situation (b).

The adaptation of a validity range V can result in a situation where $v_{max} < v_{min}$. If such situation occurs, the context agent has a non-coherent structure making it useless. To allow the system to stay in a cooperative internal state, the agent will self-destroy.

3.2.3.3. Validability and context agent creation

The system functional state requires that at least one context agent is valid at any step (see section 3.2.3). If at the end of a decision cycle, no context agent proposes to perform an action, the system is in a situation of incompetence. Two mechanisms can solve this situation: extending an already existing context agent or creating a new context agent to represent the situation.

To anticipate a situation of incompetence, the concept of *validable* is added. A validity range v_o is *validable* if and only if $o \notin [v_{min}, v_{max}]$ and $o \in [v_{min} - \Delta_{min}, v_{max} + \Delta_{max}]$. $\Delta_{min}, \Delta_{max}$ are two values (one for each bound) dynamically managed by the AVTs to control the evolution of each bound. Δ can be interpreted as the next increment of the bound. With the concept validable, context agents can propose their action in situation that are not so different to their validity domain. If the context agent proposition is selected, it has to adapt its bound by updating the nearest bound to include the current situation. This mechanism allows context agents to dynamically increase their validity ranges.

The second mechanism occurs when no context agent is valid or validable. The previously selected context agent (and not valid anymore) can create a new context agent associated with the tutor's action. The created context agent then initializes its validity range around the current position in order to represent the system's context. If $a_{tutor} = \emptyset$, two mechanisms can occur depending on the desired level of autonomy. With a *collaborative* approach, the system can ask the tutor what is the action to perform. On contrary, with a more *autonomous* approach, system will maintain the last known action, considering that the inaction of the tutor correspond to the maintaining of the current action.

Those two mechanisms, the validable concept and context creation, allow solving the third non-cooperative situation (c). On the next section, we propose to illustrate ALEX on a concrete application.

4. Experiment

We want to highlight that ALEX presents interesting properties for ambient applications that differ from the traditional approaches:

- **Genericity**: the learning process is independent of the task to perform and uses no semantic on signals.
- **Openness**: new signals can be dynamically integrated in the decision process.
- **Distributed**: self-observation allows each functionality to be autonomous while remaining cooperative.
- **Real-time**: self-organization is performed without any system downtime.

To illustrate these properties, the following experiment has been performed.

4.1. Description

A two-wheeled rover with no sensors is immerged in a 2mx2m arena composed of a blue area and a green block. An intelligent camera located perpendicularly to the arena at 2m of its center can analyses pixels to capture the position of artefacts (the blue area and the green block) relatively to the rover orientation (determined by two red markers on the rover) (Figure 1). Thus at each time step, the camera can produce four observations on the scene (Table 1). To show ALEX openness properties, the camera sends observations only if an artefact is in front of the rover. At each time step, the camera can then produce either zero, two or four observations. This means that some observation can appear and disappear.

Table 1. Camera observations

Camera observations	Symbol	Domain
Distance in pixel to the center of the green block	D_g	[-680;680]
Angle between the rover front and the green block in radian	A_g	[-2π,2 π]
Distance in pixel to the center of the blue area	D_b	[-680;680]
Angle between the rover front and the blue area in radian	A_b	[-2π,2π]

A human user performs a direct control over the rover through a 2-joystick gamepad. Each joystick controls the speed of one wheel (left joystick for left wheel and reciprocally). Speed value belongs to $[-100; 100]$ and correspond to the percentage of the maximum speed to be applied and the sign of the rotation.

The aim of the experiment is to show ALEX capacity to learn complex behaviors from the observation of the user activity. For thus, the user can perform a range of activities in the arena and it is ALEX duty to exploit this interaction to imitate the user performance.

The experiment is performed both on real and virtual world. To show both genericity and distributed properties of ALEX, an ALEX is associated to each wheel allowing each wheel to act autonomously. Each wheel is then seen as an autonomous device. The role of an ALEX instance is then to control the wheel speed by correlating the observations from the camera to the actions performed by the user. An ALEX disposes at each step of a set of observations from the camera and, if relevant, the action made by the user. One-step of decision occurs every 250ms. The exact same implementation of ALEX is used on both experiments; only network protocol will differ depending on ALEX controls either a simulated rover or a real rover. OpenCV is used on both experiments for image analysis. ALEX uses the *collaborative* approach, which means that an ALEX instance will ask to the user the action to perform when no context agent is proposing an action. The virtual experiment has been developed on Webots™ simulator using the given Boe-bot model. The real world implementation has been made with a Boe-bot rover and uses an Xbee communication protocol.

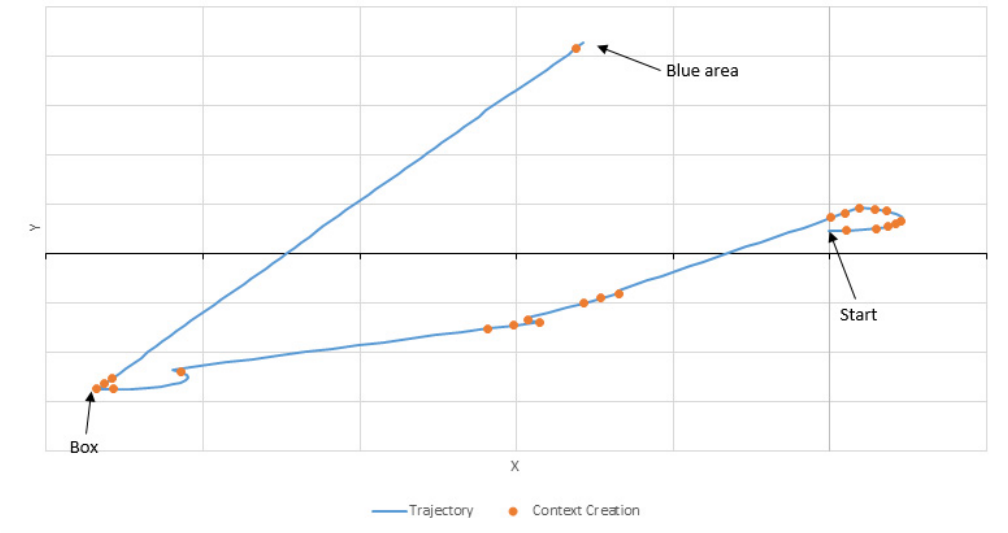

Figure 2: Rover's trajectory and context creation during a demonstration

Figure 3: On the left, the validity domain structure at the creation of a context agent. On the right, the same context agent at the end of the demonstration.

One of the activity the user can perform on the arena is a collecting task. In order to do so, solid whiskers are added to the rover to allow the rover to capture boxes. Whiskers are not movable and there is no sensor on them. They are just a physical part of the robot. Whenever the boxes are inside the blue area, boxes are moved randomly inside the arena.

4.2. Results

4.2.1. Context agent creation

Figure 2 illustrates the creation of context agents in the two ALEX instances. It shows the trajectory performed by the rover during the first demonstration of the collecting task. In this demonstration, the user takes control over the rover and drives it to collect and transport the box to the blue area.

We can observe that ALEX instances create new context agents when the user changes direction. On contrary, when the user maintains the direction of the rover, no context agents are created. Each user action is observed and each context agent determines if the current situation belongs to its own context description. If there is no such context agent, a new one is created. This illustrates ALEX capacity to detect new contexts and dynamically self-organize to integrate these new contexts in its decision process.

4.2.2. Self-adaptation of context agents

Figure 3 shows the structure of a particular context agent at its creation and at the end of a demonstration. Each line corresponds to the structure of a validity range associated to an observation. The yellow range corresponds to the *valid* range whereas the green area corresponds to the *validable* range. White boxes correspond to the current value of the signal. We observe that each validity range has its own evolution. This evolution is the result of the self-organization process. More precisely, validity ranges associated to the perception of the green block (*GreenA* and *GreenD*) are smaller than the one associated to the perception of the blue area (*BlueA* and *BlueD*). This particular context agent is valid when the block is close to the front of the rover and the rover is in front of the blue area. It is involved in the part of the activity where the rover brings back the block to the blue area.

4.2.3. Performances

To observe the capacity of the system to imitate the user performance, the user realized a 5 minutes demonstration in which 12 boxes were collected. The number of collected box by the user serves as a metric for performance comparison. The system is

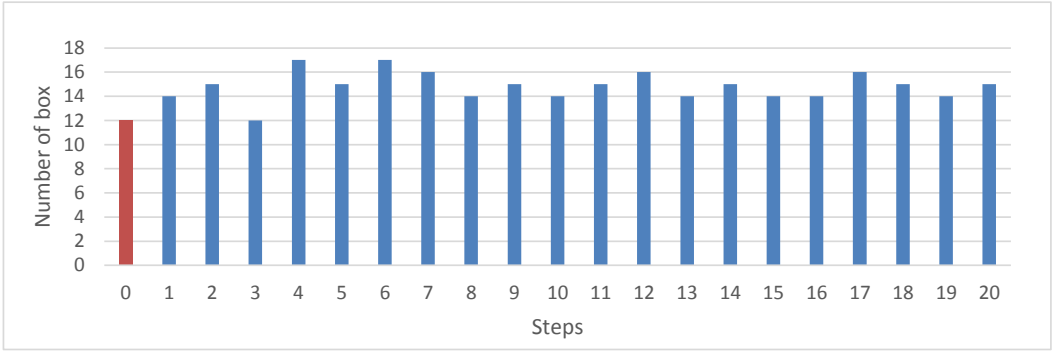

Figure 4: Number of collected boxes each 5 minutes. The step 0 corresponds to the reference score.

then let in autonomy and each 5 minutes the score is computed. During the autonomy phase, context agents have to find the most adequate action. Contrary to the collaborative demonstration phase where the user interacts with the system to teach previously unknown situations, the user never acts on the system. The figure 4 shows typical results we obtained. In the worst case, the system performs the task as well as the user does: 12 boxes are collected. However, the number of collected boxes is often better than the user's ones.

Two factors influence this result. The first one comes with the randomness of the box movements. In some case, boxes are moved farther away from the blue area and it takes more time to reach and bring back the block. The other one, more interesting, lies in the fact that the user needs more time to take a decision than ALEX does. Moreover, the user can contradict itself. This phenomenon is observable in the figure 2. At midpoint between the start position and the box position, we can observe a change in the trajectory. This change is in fact a user tele-operation mistake. Context agents corresponding to this situation will never be reelected as they correspond to a non-desired action and will then self-destroy. The learnt behavior is then "filtered" of user mistakes allowing it to perform the task more efficiently.

5. Conclusion and future work

This paper deals with the challenges of LfD in ambient systems. It presents ALEX, a multi-agent system to learn from demonstration in ambient applications. Experiments tend to show the capacity of ALEX to learn from the interaction with its user. This means concretely that a simple demonstration of the task to perform allows each multi-agent associated to all the effectors to understand autonomously what the relevant data are in order to mimic collectively what the tutor does, without any central control. Each ALEX creates and self-organizes its context agents to produce collectively a behavior that is user satisfying. The experiment has shown that two ALEX instances can cooperate without direct interaction. Moreover, it illustrates that the system is able to perform a task more efficiently than the user.

This paper is a proof of concept that multi-agent context learning can deal with the complexity of ambient systems. However, many works remain to be made. We want to consider the use of ALEX technology in more complex problems coming from industrial needs. We have a particular interest for collaborative robotic applications where workers and robots have to work collaboratively. Factories of Future (FoF) are a good illustration of such applications[16]. Evolutions of ALEX algorithm have to include the capacities to filter useless data and to discover that data are missing. Such processes could be done by adding percept agents associated to each signals who will have the responsibility to learn their utility for context agents and to help them. At last, we want both to formalize our approach and to compare its performances with other well-known learning techniques.

References

1. Kaminka, Gal A. Autonomous Agents Research in Robotics: A Report from the Trenches. In*: AAAI Spring Symposium: Designing Intelligent Robots*. 2012..
2. Chung, J. Ambient robotics for meaningful life of the elderly. *Advanced Construction and Building Technology for Society*, 2014, p. 43.
3. Russel, S., Norvig, P.*, Artificial Intelligence: A modern approach*. Prentice-Hall, Englewood Cliffs, 1995, vol. 25.
4. Hagras, H., Callaghan, V., Colley, M., & al. Creating an ambient-intelligence environment using embedded agents. In: *Intelligent Systems, IEEE*, 2004, vol. 19, no 6, p. 12-20.
5. Kearns, Michael J., Schapire, Robert E., Sellie, Linda M. Toward efficient agnostic learning. *Machine Learning*, 1994, vol. 17, no 2-3, p. 115-141.
6. Argall, Brenna D., Chernova, Sonia, Veloso, Manuela, & al. A survey of robot learning from demonstration. In: *Robotics and autonomous systems*, 2009, vol. 57, no 5, p. 469-483.
7. Billard, A., Calinon, S., Dillmann,R., Schaal, S. Robot programming by demonstration. In: *Springer handbook of robotics*. Springer Berlin Heidelberg, 2008. p. 1371-1394.
8. Billing, Erik A., Hellström, Thomas. A formalism for learning from demonstration. *Paladyn*, 2010, vol. 1, no 1, p. 1-13.
9. Dautenhahn, Kerstin, Nehaniv, Chrystopher L. *The correspondence problem*. MIT Press, 2002.
10. Verstaevel, N., Régis C., Guivarch, V., Gleizes, M.P., Robert, F. Extreme Sensitive Robotic: A Context-Aware Ubiquitous Learning. In*: Proceedings of the 2015 International Conference on Agents and Artificial Intelligence*, 2015, vol. 1, p. 242-248.
11. Guivarch, V., Camps, V., Péninou, A., & al. Self-Adaptation of a Learnt Behaviour by Detecting and by Managing User's Implicit Contradictions. In: *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. IEEE Computer Society, 2014, vol. 3, p. 24-31.
12. Gleizes, Marie-Pierre. Self-adaptive complex systems. In: *Multi-Agent Systems*. Springer Berlin Heidelberg, 2012. p. 114-128.
13. Capera, Davy, Georgé, Jean Pierre, Gleizes, M.-P., & al. The AMAS theory for complex problem solving based on self-organizing cooperative agents. In*: Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on. IEEE, 2003. p. 383-388.
14. Bonjean, N., Mefteh, W., Gleizes, M. P., & al. ADELFE 2.0. In: *Handbook on Agent-Oriented Design Processes*. Springer Berlin Heidelberg, 2014. p. 19-63.
15. Lemouzy, S., Camps, V., Glize, P. Principles and properties of a mas learning algorithm: A comparison with standard learning algorithms applied to implicit feedback assessment. In: *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on. IEEE*, 2011. p. 228-235.
16. Sicilliano, Bruno, Caccavale, Fabrizio, Zwicker, Ekkehard, & al. EuRoC-The Challenge Initiative for European Robotics. In: Proceedings of *41st International Symposium on Robotic*s, VDE, 2014. p. 1-7.