Learning and Adaptation in Physical Agents

Sandor M Veres * **Aron G Veres** **

* *University of Southampton, UK, s.m.veres@soton.ac.uk*
** *SysBrain Ltd, Birmingham, UK, Email:aron@sysbrain.org*

Abstract: Learning and adaptation is fundamental for autonomous agents that operate in a physical world and not a computer network. The paper is providing a general framework of skills learning within behaviour logic framework of agents that communicate, sense and act in the physical world. It is advocated that playfulness can be important in learning and to improving skills of agents.

Keywords: Intelligent physical agents, formal methods, learning, adaptive control modelling for control, .

1. INTRODUCTION

In the field of engineering we want autonomous agent that control machines [10, 4] to be (1) capable of solving and executing complex problems in the physical world and we also want them to be (2) reliable. Reliability in essence means that we know how they behave. Reactive agents are good for this as we can prescribe or let them learn the rules by which they behave. Deliberative agents using modelling and complex decision making are however difficult to verify for all physical environments and hence their reliability is more difficult to prove. This apparent dilemma is attempted to be partially solved by an agent architecture in this paper that prescribes a clearly defined behaviour logic for reliability while retaining the ability of learning, adaptation and complex problem solving by modelling. The architecture is verifiable by design.

2. BEHAVIOUR DESCRIPTIONS

A language $\mathbf{\Lambda}_{ABL}$ of temporal *agent behaviour logic* is defined over a set of atomic formulae $M_o = \{p, q, ...\}$ called *operational modes* (OMs) as follows:

$$\phi = p|\neg\phi|\phi \vee \phi|\phi \wedge \phi|\Box\phi|\Diamond\phi|\phi \rightharpoonup \phi|\top|\bot \tag{1}$$

For each operational mode in M_o there is an activity dynamics associated by the *activity function* defined as

$$A : M_o \mapsto F_b \tag{2}$$

where F_b is a set of feedback loops between the agent's actuators and a part of the agents internal or external environment (internal is for instance for iterative refinement of plans of future actions).

There are three important temporal functions defined over the set F_b. The first one is the Boolean temporal *activation function* $a : F_b \to \{0, 1\}$ and the second one is the *activity value function* $v : F_b \to [-1, 1]$ and the third one is the timeout function $t : F_b \to [0, \infty]$. The activation function provides a semantics for the logic of operational modes (OMs) as the a can be used to evaluate any temporal logic behaviour formula through the activity functions associated with OMs.

Definition 1. A logic formula is called a *simple behaviour* if it only contains the operations $\vee$, $\wedge$ and $\rightharpoonup$.

A simple behaviour defines parallel activities connected by ($\wedge$), sequential activities connected by $\rightharpoonup$ and activity options connected by $\vee$ relations. For instance the $p \vee (q \wedge r) \vee (s \rightharpoonup u \rightharpoonup w)$ can mean that either the p operational mode (OM) is on or the q and r OMs are simultaneously on or the OM s is first on then followed by OM u which is followed by w. When w stops operating then either p must start, or q and r or s needs to restart again.

Definition 2. A *1st-order* autonomous physical agent $\mathbf{A}$ is defined by the tuple $\mathbf{A} = \{M_o, A, F_b, a, v, t, B\}$ where B is a simple behaviour in terms of the OMs in M_o.

A 1st-order physical agent (PA) defines its semantics in terms of its activity function a. At any moment of time its behaviour formula B can be evaluated in terms of a. Note that satisfaction of B at any time instant does not mean anything about the success or reliability of the agent, it merely says that at any time the agent will activate OMs in accordance with satisfying formula B . For any A and B the $A \rightharpoonup B$ is defined true if either A or B holds true and in temporal sense B follows A. Operations of $\vee, \wedge$ are defined as usual as "or" and "and" between operational modes.

Definition 3. A *1st-order* autonomous physical agent $\mathbf{A}$ is is called *consistent* if B is true as a temporal logic formula, i.e. $\Box B$ evaluates to true using a.

The semantics of behaviour logic formulae is that they take on true 1 or false 0 values during the course of time via the mapping a. $s \rightharpoonup u$ means that s is followed by u in time, hence temporal logic is needed instead of propositional logic (if only $\vee, \wedge, \neg$ were used then propositional logic would suffice).

Function a is evaluated over the temporally changing operational modes F_b and expresses the fact that the agent runs the algorithms of F_b. The qualification of success of operations F_b is expressed by function v, which means poor performance for low positive values, very good performance for values near to 1, instability or totally unacceptable performance for small negative values of v and damaging or dangerous performance for v close to -1.

Definition 4. A *1st-order* consistent autonomous physical agent $\mathbf{A}$ with simple behaviour formula B is called *safe* if it always evaluates an active OM with $v > 0$. $\mathbf{A}$ is called *reliable at level* $\epsilon > 0$ if the active OMs are always evaluated to $v > 1 - \epsilon$ eventually.

The $v > 0$ condition being defined at any time instant does not mean that $v > 0$ is only dependent on feedback loop data at that time instant. $v > 0$ is typically a function of a control criterion (or self-made goals and external rewards) that is obtained from a sequence of past feedback input-output data.

Operational safety depends on the actual interaction of the agent with its environment. A lot can be achieved for safety by simply altering the a switching function so that if an OM approaches the $v < 0$ region then the agent is switched to another OM that is used to rescue the situation. Whether that will help to achieve overall objectives is another matter and is part of the overall performance evaluation of the agent. Automating a consistent switching mechanism is the topic of the next section.

3. SWITCHING BETWEEN OPERATIONAL MODES

The activity function a of a 1st-order PA is changing over OMs as the OMs progress. There are the following practically important cases to consider:

(1) An active OM is successfully completed at a required level reliably.
(2) An active OM is not successful at the required level (e.g. $v < 0.5$) but it still operates safely.
(3) An active OM is timed out.
(4) An active OM is being aborted by another active OM.

The interaction of an agent with its environment under a single operational mode is normally the topic of control engineering. This control engineering problem can however be widened with the realtime variable choice for actuators and sensors signal to be used in the feedback loop.

The objective of this paper is to establish safe operational mechanisms of autonomous physical agents that learn. The previous section introduced the important concept of consistency for 1st-order agents. A further step towards safe operation and good performance is to analyze the conditions of successful operation and bring them together with the above logical framework. To keep the new formalism to a minimum, any $f \in F_b$ will be associated with an initial condition $I(f, M_f)$ that is a 0 or 1 Boolean valued relation function between the agent and its environment.

Definition 5. We say that the *adaptation condition is ϵ-satisfied* by an operational mode f if it holds that whenever $I(f, M_f)$ is satisfied and the agent has f active, the performance $v(f)$ is guaranteed to converge to and stay inside $[1 - \epsilon, 1]$ within time period $t(v)$ under uncertain environmental model M_f.

A biped robot may be able to start and walk nicely ($v(f) \rightarrow [1 - \epsilon, 1]$) if not starting from a lying or fallen-over position but if it already stands reasonably upright ($I(f, M_f)$), even if perhaps a rucksack has been placed on its back. The latter means that under some initial condition the operational mode of the walking of the robot is adaptive. If the condition $I(f, M_f)$ of walking is not satisfied then the robot may decide to switch to another operational mode f_1, meaning for instance that the robot is "trying to stand up" first, an action that is an operational mode itself.

Individual OMs of the agent can be tested by formal analysis and practical testing. Hence the above analysis highlights the relevance of enforcing such an a on agent behaviour that starts any $f \in F_b$ under condition $I(f, M_f)$ being satisfied that leads to eventually $v(f) > 1 - \epsilon$.

Lemma 6. A *1st-order* consistent autonomous physical agent $\mathbf{A}$ is reliable at level $\epsilon > 0$ if the following two conditions are satisfied:

(a) All operational modes $f \in M_f$ satisfy the adaptation condition at level ϵ.
(b) The activity function a is such that whenever an $a(f)$ becomes 1 for an $f \in F_b$ then $I(f, M_f)$ is satisfied.

Proof. Straightforward from the definitions: as all operational conditions are adaptive, and start from correct initial conditions, the performance function v will rise above $1-\epsilon$ for any operational mode within its time limit. □

3.1 Example 1

An unmanned light aircraft can have the following tasks:

- $M_1 \rightarrow F_{p1}$: *Warming up and control tests.*
- $M_2 \rightarrow F_{p2}$: *Checking mission instructions.*
- $M_3 \rightarrow F_{p3}$: *Planning flight.*
- $M_4 \rightarrow F_{p4}$: *Taking off.*
- $M_5 \rightarrow F_{p5}$: *Taking mission related pictures and measurements.*
- $M_6 \rightarrow F_{p6}$: *Controlling the plane at normal altitude while following mission path.*
- $M_7 \rightarrow F_{p7}$: *Deciding to abort mission.*
- $M_8 \rightarrow F_{p8}$: *Searching for emergency landing area.*
- $M_9 \rightarrow F_{p9}$: *Planning flight to emergency area.*
- $M_{10} \rightarrow F_{p10}$: *Emergency landing.*
- $M_{11} \rightarrow F_{p11}$: *Planning normal return flight.*
- $M_{12} \rightarrow F_{p12}$: *Normal landing on return.*
- $M_{13} \rightarrow F_{p13}$: *Controlling the plane in emergency.*
- $M_{14} \rightarrow F_{p14}$: *Modelling flight path followed until current time.*
- $M_{15} \rightarrow F_{p15}$: *Writing and sending mid-flight report on conditions on board.*

The following behaviour formula is an example:

$$\begin{aligned} B_2 = ((M_1 \rightharpoonup M_2 \rightharpoonup M_3) \vee (\ M_4 \rightharpoonup M_6) \vee M_5\ )\vee \\ \vee(M_{13} \wedge (M_7 \rightharpoonup M_8 \rightharpoonup M_9))\vee \\ \vee(M_4 \rightharpoonup M_6 \rightharpoonup M_{12})) \wedge M_{14} \end{aligned} \tag{3}$$

The dynamical control, planning and emergency challenges and the algorithms are more complex here than in the lawnmower example. Still the same principles of using the a, v, t functions can be used to assess reliability and performance. The significance of adaptation and learning is even more emphasized here.

3.2 Example 2

Assume that a garden robot has the following mission: either (1) mow the lawn or (2) turn on the watering system or (3) recharge its own batteries or (4) empty the grass from its container to a prescribed dump site (5) report to a human operator for maintenance. Within each of these tasks there are several OMs to be executed consecutively:

- $M_1 \rightarrow F_{p1}$*: Mowing.*
- $M_2 \rightarrow F_{p2}$*: Planning of mowing.*
- $M_3 \rightarrow F_{p3}$*: Watering.*
- $M_4 \rightarrow F_{p4}$*: Planning of watering.*
- $M_5 \rightarrow F_{p5}$*: Empty grass container.*
- $M_6 \rightarrow F_{p6}$*: Planning route to charging point.*
- $M_7 \rightarrow F_{p7}$*: Recharge.*
- $M_8 \rightarrow F_{p8}$*: Decide on and request maintenance.*
- $M_9 \rightarrow F_{p9}$*: Write problems report.*
- $M_{10} \rightarrow F_{p10}$*: Map building.*
- $M_{11} \rightarrow F_{p11}$*: Self modelling of hardware for diagnostics.*
- $M_{12} \rightarrow F_{p12}$*: Modelling of past mowing, watering and maintenance work completed.*
- $M_{13} \rightarrow F_{p13}$*: Planning for emptying grass container.*
- $M_{13} \rightarrow F_{p14}$*: Idle (standby).*

Out of these M_1, M_3, M_5, M_7 are feedback-loop based operational modes that need sensing and control of actions accordingly. M_2, M_4, M_6 are on the other hand OMs that need algorithms working on models only and do not need sensing or actuation. OMs M_{10}, M_{11} and M_{12} need sensing only and algorithms that build models from sensor data. Decisions by M_8 are based on internal models and may result in sending a message to the human operator after writing report M_9 on the problems that may need maintenance.

A behaviour logic formula that the autonomous lawnmower needs to satisfy can for instance be

$$\begin{aligned} B_1 = (\ (M_2 \rightharpoonup M_1 \rightharpoonup M_{13} \rightharpoonup M_5) \vee (M_4 \rightharpoonup M_3) \vee \\ \vee (M_6 \rightharpoonup M_7)\ ) \wedge (M_8 \rightharpoonup M_9) \wedge M_{11} \wedge M_{12} \wedge M_{10} \wedge M_{14} \end{aligned} \tag{4}$$

The a switch must be such, as decided within each operational mode, that the total formula B must always hold true. This means that the parallel modelling and maintenance monitoring operational activity carry on while one of the mowing, watering or recharging tasks are executed. Despite the essentially reactive behaviour the lawnmower has the ability of interpretation of the environment and planning while strict discipline of behaviour code is maintained. Based on sensing or assessment of algorithmic results, the evaluation of v is constantly carried out for each operational mode.

Now the reliability at level ϵ is achieved if all feedback and open-loop OMs are proven to work under uncertain models of the environment and the initial conditions are always achieved when switching to a new operational mode. To achieve the necessary initial conditions the OM algorithms need careful action around the switching points. When the physical and control algorithmic work on the lawnmower robot has been completed, then the satisfactory nature of a, v, t can be formally tested. As this may be difficult in practice, adaptation and learning in the OMs is therefore vital to reduce development effort and to achieve level-ϵ reliability of the autonomous lawnmower.

Both examples suggest that logical consistency provided by a is a fundamental requirement. Beyond that, safety and performance of the system depends on the quality of the OM algorithms to provide suitable v functions. Whatever is achieved and guaranteed in terms of the v, there is scope for *further formal analysis and alterations to be made* to enforce safety switches in face of undesirable performance in some operational mode. This is the topic of automated adaptation and learning.

4. ADAPTATION AND LEARNING OF 1ST-ORDER AGENTS

The principle is that adaptation is performed by

- adjusting feedback controller parameters or NN weights during the operational mode
- associating feed-forward actions, that were successful in the past, with short term planning and using them in control.

Learning can be restricted to OMs that include feedback based interaction with the environment, for instance path following of the law mower, its approach and connection to the recharging point, or the flight control parameters of the plane under various operating conditions. Measurement of performance and success is crucial in feedback loops of interaction with nature so that improvements can be made by learning. Performance is measured by the evolution of the v-functions during the execution of an OM. The objective of adaptation and learning is to increase v and to make it converge faster. This is only achieved if the agent successfully adapts its OMs to varying environmental circumstances. In this section we briefly review the main available techniques for learning in each operational mode.

4.1 Tuning parametric feedback/feedforward controllers

On-line parametric feedback tuning of an $f_i \in F_b$ is one of the fastest learning methods. The v can be defined as a monotone function of a control performance criterion and some parameter vector θ_i of the feedback/feedforward (FB/FF) controller is to be tuned. The principle of tuning is to compute the gradient direction of the cost function using online measurement data of sensors and actuators and hence move the parameter θ_i uphill to increase the performance measured by v [9].

This approach assumes that some good controller structure is available from a priori analysis of the physical problem. Also relatively good initial θ_i is needed that already ensures stable feedback under the conditions $I(f, M_f)$. Although a priori design is essential, this learning mechanisms is the fastest from the ones considered in these sections. This is not surprising because this approach to learning benefits from human intelligence and understanding of models of possible operational circumstances. On the other hand there are practical limitations to how much a human designer can model the dynamical details and is able to think of all possible changes against which adaptation is needed. That is why NN (neural networks), reinforcement learning, model-based control and associative learning are of great importance.

4.2 Neural network learning

Artificial neural networks can be used to tune FB/FF (feedback/feedforward) control action using multilayered perceptron, RBF and recurrent networks, etc.[2]. These schemes can also be combined with associations of operating conditions. So control under an $f_i \in F_b$ can be made dependent on past associations of measurements from all sensors and successful control attempts. This way neural networks can be complemented by associative learning. The resulting combined associative NN and dynamical control NN together form a self-organizing controller[1]. Self-organization appears as associations will determine which NN structures are formed.

The disadvantage of this approach is that training may take a long time. The greatest advantage is robustness and potentially superior performance over parametric methods as NN-based tuning may be able to make use of control opportunities that were not discovered by the human engineer designing the autonomous system. NN learning assumes that there is plenty of time and opportunity to practice operational modes. A most successful and innovative way to speed up NN tuning is to use it as complementary to model based feedback loop design to enhance the performance of model based methods.

4.3 Learning by reinforcement

When some prediction system is maintained on the effect of considered control actions then exploratory and exploitative (greedy) actions can be taken in reinforcement learning. In addition to v that is a realtime performance measurement, we can introduce a performance estimate (=value function) $\hat{v}$ that is updated as the measured state of the system evolves. Successful performance will then propagate into desirable state and learning slowly progresses. Temporal difference learning adjusts the anticipated $\hat{v}$ through exploratory and greedy actions [6].

As the size of the state-space can be large reinforcement learning can help coping with complexity. In reinforcement learning $f_i \in F_b$ can rely on strongly discretised description of a not too large state space. Applied directly, continuous state dynamical control is difficult to learn by reinforcement learning due to complexity. Improved reinforcement learning can be used with regard to the selection of the control initial parameters given environmental circumstances, i.e. initializations that later lead to success or failure under perceived environmental conditions. Using this way reinforcement learning can be very useful for continuous state dynamical control.

These methods assume that the autonomous system has plenty of time and opportunity to practice its skills in terms of OMs.

4.4 Model predictive control - model learning

One of the most powerful methods of control is through the autonomous system modelling its environment, planning its sequence of actions and immediately executing the first action planned (or first few). Then it senses changes in the situation and plans again and executes the first action in the new plan again under slightly changed circumstances. This is *receding horizon control* by the agent. For the effectiveness of this generic method, the autonomous agent needs to maintain sufficiently good quality models of its environment. Also this online model should possibly be supported by a redundant set of measurements to make the data secure. A useful method is to make this realtime modelling adaptive in the sense of (1) goal selection (2) its use of the set of i/o variables in various control tasks and (3) in terms of the realtime model maintained.

5. THE ROLE OF PLAY FOR 1ST-ORDER AGENTS

Why do children play? Why do kittens play? Why do adults play games, and solve crosswords? Playing provides opportunity to practice skills. From the previous sections it is clear that one aspect of learning is to gather data for learning under non-dangerous circumstances. If a 1st-order agent only executes live-mission tasks then it is likely that a huge amount of development effort will be needed to make it to operate safely, all OMs need to be performing very well from the very start and also at any switch of operational modes the initial condition must be strictly kept.

An alternative is to build a basic structure (using parametric controllers, NNs, self-organizing NNs, reinforcement learning structures, adaptive modelling, etc.) of each operational mode of a 1st-order agent and then endow it with the ability to randomly play with the purpose of improving its skills (=operational modes). This section provides a solution by adding a "play operational mode" to the agents behaviour logic.

Let B be the simple behaviour logic formula of a 1st-order agent $\mathbf{A}$. An extended formula $B_p = B \vee M_p$ is obtained by adding an operational mode M_p that takes now a supervisory role. When functional, the job of M_p is to monitor performance of each operational mode under various environmental circumstances and randomly choose from a set of playing activities and execute them by interfering with the normally used activity function a . Playing activities are designed such that they do not interfere with the overall final goals of a mission and they are always obtained as a modified portion of the B. This is achieved by suitable changes in the switching of a.

For instance in the above examples of the lawnmower, some playing activities can be obtained as follows: (a) practice docking for recharging; (b) practice fast and slow mowing on rough ground or high grass; (c) practice finding the boundary of the garden lawn; (d) practice activating the watering system and sensing of how it works. The purpose of the practice is not merely to repeat tasks as that would be useless: its purpose is to fine-tune the OMs of the 1st-order autonomous system, i.e. to adapt the discrete and continuous control parameters in OMs. For that each OM must have a tuneable structure with learning mechanisms.

For the light autonomous aircraft some learning activities can be: (a) Taking off, doing a small round and landing straight away under various wind conditions; (b) practicing turns during normal flight while keeping flight path essentially the same (c) practicing taking photos by trying to keep the plane steady during exposure (d) practicing collision avoidance when noticing an object potentially in the way (needs human cooperation), etc.

Playing activities for M_p can be preprogrammed by the engineer developing the agent or M_p can also be generated automatically from B. Given the very simple nature of 1st-order agents, a straightforward method is that the engineer designs a series of playing activities for the agent. The task of M_p is then to seek out opportunities when these can be played, or depending on learning skills, to activate them. When playing activities are executed learning should take place automatically as all operational modes should be programmed with learning ability.

6. A LABORATORY FORMATION CONTROL EXAMPLE

Three 5DOF laboratory satellite models in the authors laboratory at the University of Southampton are used to illustrate the learning architectures of agents as described. Previous work related to this facility is described in

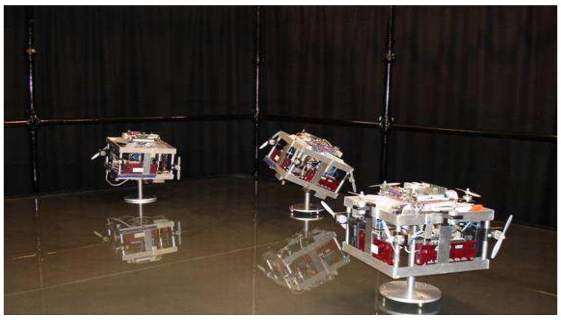

Fig. 1. The laboratory satellite models used.

[3, K. K. T. Thanapalan, 7, 5, S. M. Veres and Lincoln, 8]. Each satellite model (satmod) is controlled by an agent belonging to a class named *satbrain* that prescribes that all three satellites are controlled by the same agent architecture. The programming differences only arise from different hardware connections. Each satmod has a 3D solid state gyroscope fixed to its frame that provides three angular rate measurements about each of the x-y-z axis and three acceleration values in each axis direction as fixed to the frame. On the metrology frame, surrounding the table where the satmods operate, there are 12 cameras that can locate any point in the space of the table by 10mm precision. These cameras are operated by 4 camera agents of class *intellicam*, each of which operate 3 cameras. The class of an agent fully determines its sensing, control codes and its decision making. Differences between agents of the same class arise from their different hardware connections, their different past and hence different parameters learned in their acquired skills. Humans can communicate with the team manager through commands that describe required formations from the satellite team. A formation is defined by positions (2D) and attitudes (3D) of each satellite. Here we focus on describing the operational modes (=skills) related to learning.

6.1 Learning operational modes

There are four agent classes used for he 9 agents controlling the 3 satellite frames: *intellicam, managecam, satbrain0* and *team-manager*. In the current implementation each of these except the team manager has adaptive operational modes.

The agent class *intellicam* has the behaviour formula

$$idle \vee identify.markers \tag{5}$$

This means a single operational mode *identify.markers* that has to tune and re-tune the parameters of the marker recognition procedure. This mainly means the tuning of RGB colour code intervals. The procedure includes the sending of the data to the camera manager upon request through WLAN. The data sent are pixel positions of markers on the satellites. The agent class *managecam* has the behaviour logic formula defined by

$$\begin{gathered} idle \vee feedback(estim.attitude\ \&\\ estim.pos\ \&\ send.estims) \end{gathered} \tag{6}$$

that means three operational modes (OMs): *idle, estimate.attitude, estimate.position.* Each of these are associated with a (1) Boolean variable whether the operational mode is active and a (2) code that executes the actions to be taken. The OM *estimate.attitude* consists of geometric computations that has the built in adaptation ability to missing marker data and performs data fusion if position estimates are irrealistic relative to previous estimates.

The agent class *satbrain0* has the behaviour logic formula B_s defined by

$$\begin{gathered} idle\ \vee\ operate(monitor(model\\ \vee reconfigure(diagnose\ \rightharpoonup\\ choose.controller)))\\ \&\ (plan.route\ \rightharpoonup\ track.route)) \end{gathered} \tag{7}$$

This means that when the satmod operates then it simultaneously monitors the situation and either plans or executes a movement (tracking of plan). Monitoring means constant modelling or switching to reconfiguration if modelling identifies a problem. The learning part is the tracking controller which is measured by performance functions v. Timeout function t is also defined and activates learning mechanisms for control. Learning in OM *track.route* is based on adaptive multi-variable control. This adaptive controller adjusts controller parameters. Reconfiguration is only applied to this controller if drastic hardware deficiencies are identified by the continuously performed dynamical modelling.

6.2 Fault tolerant self-reconfiguration

In the present setup the reconfiguration is based on modelling, conclusion rules and a fixed set of reconfiguration possibilities determined in advance. Possible faults can be if a thruster does not work near its nominal gain or if inertia based measurements are needed as the vision system did not provide reliable data that are consistent with predicted regions of position and attitude.

6.3 Playfulness definitions

For playful behaviour control the behaviour formula B_s is of *satbrain0* is augmented by an operational mode

$$\begin{gathered} B_{play} = idle \vee (plan.random.movement\ \rightharpoonup\\ activate.OM) \end{gathered} \tag{8}$$

As the dynamical control part is adaptive it improves with practice, hence the playful behaviour mode activates operational modes that can learn by practice.

Reliability is enhanced exactly because of improved performance levels of controllers working in operational modes.

7. SOFTWARE

Software that can be used to implement the above agent architectures, to launch and operate them in a MATLAB based environment on a network of computers, can be found at sysbrain.com. Libraries of operational modes and MATLAB m-files for perception in autonomous control systems, can be found at sysbrain.org.

8. CONCLUSIONS

This paper introduced a simple autonomous physical agent architecture based on behaviour logic, when combined with learning algorithms, can achieve high levels of safety and reliability. The capability of higher levels of abstraction are not required to be able to exhibit highly adaptive, planning based behaviour that is normally associated with higher levels of intelligence. To make PAs robust in natural environments, where unexpected circumstances often occur, it is proposed that playful behaviour is essential at a training phase of agents. It has been illustrated that adding playful behaviour is relatively simple in the behaviour- formula-based agent architecture proposed.

REFERENCES

[1] J. H. Andreae. *Associative Learning.* Imperial College Press, London, 1998. ISBN ISBN 1-86094-132-X.

[K. K. T. Thanapalan] E. Rogers S. B. Gabriel. K. K. T. Thanapalan, S. M. Veres. Fault tolerant controller design to ensure operational safety in satellite formation flying. *Proc. 45th IEEE Conference on Decision and Control (CDC), December, 2006, San Diego, USA.*

[2] Y. H. Kim and F.L. Lewis. *High-Level Feedback Control with Neural Networks*, volume 21 of *Robotics and Intelligent Systems.* World Scientific, Singapore, 1998. ISBN ISBN 981-02-3376-0.

[3] N. Lincoln and S. M. Veres. Components of a vision assisted constrained autonomous satellite formation flying control system. *International Journal of Adaptive Control and Signal Processing*, (2), 2007.

[4] A. M. Meystel and J. S. Albus. *Intelligent Systems: Architecure, design and Control.* Wiley Series on Intelligent Systems. John Wiley and Sons, Inc., New York, 2002. ISBN ISBN 0-471-19374-7.

[5] E. Rogers S. M. Veres, S. B. Gabriel and D. Q. Mayne. Analysis of formation flying control of a pair of nanosatellites. *AIAA Journal of Guidance, Control and Dynamics*, pages 971–974, 2002.

[S. M. Veres and Lincoln] S. B. Gabriel S. M. Veres and N. K. Lincoln. Facility for satellite formation flying system verification. *Proc. 9th SESP ESA Conference.*

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning - An Introduction.* Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 1998. ISBN ISBN 0-262-19398-1.

[7] S. M. Veres. Autonomous formation flying of satellite robots: The mechanical control layer. *Proc. Conference on Autonomous Robotic Systems (TAROS'06),September 2006*, CD, 2006.

[8] S. M. Veres and J. Luo. BDI agent architectures for autonomous control. *Proc. IEEE Conference on Conference on Decision and Control (CDC), December 2004.*, CD, 2004.

[9] S. M. Veres and D. S. Wall. *Synergy and Duality of Identification and Control.* Taylor & Francis, London, 2000.

[10] M Wooldridge. *An Introduction to Multiagent Systems.* John Wiley & Sons, Chichester, 2002. ISBN ISBN 0-471-49691-X.