Proceedings of the International Symposium on AI-Inspired Biology, Jackie Chappell, Susannah Thorpe, Nick Hawes and Aaron Sloman (Eds.), at the AISB 2010 convention, 29 March – 1 April 2010, De Montfort University, Leicester, UK

Modelising H-CogAff global alarm system improves robot performance

Mary Felkin and **Yves Kodratoff** [1]

Abstract. We developed a system which purpose is to obtain a robot able to emulate the strategies used by a subject facing a problem-solving task. We have been able to solve this problem within a strongly constrained setting in which the subject's strategy can be induced. Our solution encompasses the solution of another problem, namely how to close the raw-data/trained-system/raw-data loop. An important aspect of inducinsubjectan strategies was becoming able to recognise changes of strategy. The H-CogAff from which our control architecture is inspired (though we do not yet attempt to completely emulate it) provides a handy trigger for strategy changes: the global alarm system. We show that implementing it improves the robot's performance.

1 Introduction

There are many abstract definitions of human intelligence and not enough engineering blueprints of it. Paradoxically, working models abound all around us: human beings. So we teamed up with psychologists to study and emulate subjects in a problem-solving context. Though the different relevant behaviours of human beings in a limited environment and with a fixed goal can be identified and simulated, the strategies which go on in the higher levels of the "control architecture"[2] which structure these behaviours are harder to induce: doing so requires a careful step-by-step approach. To guide us in exploring the as yet incompletely chartered territory of the subject "control architecture" we chose the H-CogAff map. This turned out successfully so we are now taking a step further and tying our model more closely to H-CogAff by implementing a global alarm system.

From an AI/Robotics standpoint, when a mechanism which mimics a theoretical human thought process improves the performance of the robot, this validates the theory.

2 Related work

A large amount of work has been done in the field of robot learning by imitation, a relatively new (about twenty years old) field of research, see for example [5], [10] and [19]. This field takes inspiration from a wide range of disciplines, including psychology, biology, neurobiology, etc. [2], [4], [9] and [6]. An example among others of the necessary multidisciplinarity is [3] who propose a mathematical solution to the correspondence problem, which originally comes from animal psychology: they formalise the correspondences by giving mapping matrices to link agents with different morphologies. Other research papers present architectures which are less biomimetic, for example [7] who present an architecture for extracting the relevant features of a given task and then generalise the acquired knowledge to other contexts. They demonstrated the effectiveness of their architecture by implementing it on a humanoid robot learning to reproduce the gestures of a human teacher, not all architectures are inspired by cognitive sciences, our approach is just one among many. A formal definition of plan recognition can be found in [15]. Close to our work but using a different mathematical framework are Bayesian methods: [24] focuses on the higher levels given a set of basic robotic actions and [20] on real-time Bayesian learning. [14] compare several approaches and discuss imitation learning in virtual reality.

3 The raw-data/trained-system/raw-data loop

This loop, which to the best of our knowledge we were the first to close, enables for example a robot to learn from another robot. When domestic androids will reach the mass production stage few users will be happy to train their robot from scratch. Generalising human strategies from many examples of tasks such as clearing the table after a meal in different settings will enable the designers to abstract these strategies from their particular context. A simulated robot could then enact them (at first for testing purposes) in a much wider variety of circumstances, including unlikely ones (for example in cases where the dish washer would be in the cellar). The simulated robot could clear tables in a million different simulated houses. So subsets of cases in which the robot fails could be identified and a new stategy learnt for these cases.

Once the robot's behaviour becomes fully adequate for the task, its recorded actions would provide as much learning data for subsequent robots as would be required for robust generalisation: Abstracted strategies are also abstracted from body built because our method oversteps the correspondence problem[3] so the same strategy could be implemented by many different androids.

This allows incremental additions to the robot's capacities because as we will see logs can be combined. The android can learn by imitating its user that the blue glasses go into the glass cabinet and smoothly add this knowledge to its previous strategy.

Our raw-data/trained-system/raw-data loop is a loop starting with the behaviours of several subjects and their analysis and interpretation in terms of human observable actions. This leads to the definition of the strategies used by the subjects (including inefficient ones), the interpretation of the human observable actions in terms of movements of the robot and to the definition of what is a "robot

[1] CADIA, University of Reykjavik, Iceland and LRI, Université Paris-Sud, France

[2] Strategies are considered to be a subset of meta-management processes.

[3] In robotics, the "correspondence problems" refers to the problems which arise from the fact that no (existing) robot is built exactly like a human. A humanoid robot may have a morphology and physical capacities which are quite different from these of its instructor [3]. In our example the simulated "teaching robot" and the "learning robots" may all be built in different ways

strategy" in terms of human strategies. Our loop is closed with a programming language enabling us to implement these robot strategies in new settings, either by hand or automatically from the log of the subjects' actions in the original settings. When the induced strategies are enacted, they become observable in the same way as the subject's strategies were observable at the beginning of the loop.

4 Experimental settings

Our final goal was not to replicate the behaviour of individual volunteers but to "understand" their underlying strategy and to become able to reproduce it in new surroundings.

In a sequence of psychological experiments, blindfolded human volunteers explored a maze[4] in search of a "treasure" (a bottle of water) and, while doing so, they expressed their search strategy[5] by sequences of perception-actions pairs, which were recorded. Perception here was limited to touch, which could be observed on the videos. Actions were limited to moving in the maze, touching objects and picking up the treasure, so these could also be observed. Fig. 1 illustrates the process of capturing the volunteers movements in our simulator and then recording the databases of observable actions.

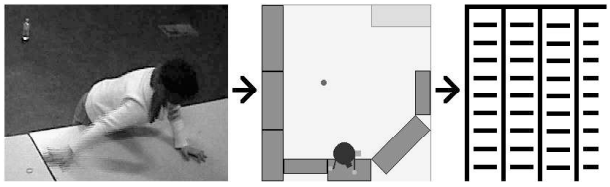

Figure 1. From videos to databases

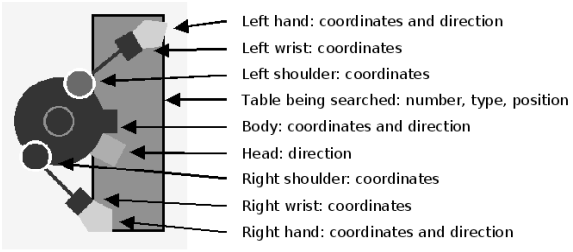

Figure 2. Some of the observable variables

The psychologist [13] and the mixed team [23] showed that the volunteers had several different goals which they combined through some thought process akin to multi-criteria optimisation to mentally construct and evaluate their behaviours. On top of their given goal, finding the treasure, their most often used strategies included the goals of not getting lost, of not exploring the same place twice, of not bumping into obstacles, etc. We performed a detailed analysis, including a digitalisation, of the videos showing the behaviour of 10 of these volunteers, called G1_1, G1_2, G1_3, G3_1, G3_2, G4_1, G4_2, G4_3, G7_1 and G7_2 in the following. We thus could run a close[6] replicate of their behaviour in our system and analyse it.

[4] The mazes were not virtual, they were built with rows of tables and sometimes cupboards and radiators in a large schoolroom.

[5] Our use of the expression "search strategy" here does not imply the volunteers were searching according to an explicit plan. Randomly searching through the maze is also a "search strategy", and so is "not searching at all".

5 A step-by-step analysis

Automatically extracting from a database the strategies used by subjects in a problem-solving situation takes more than a good preprocessing and then running the database through the appropriate data mining algorithm. To go from the database of observables to the strategies, we had to define a middle ground. Fig. 3 models the human's cognitive processes as a very simplified version of the H-CogAff (Human Cognitive Affects, [21]) model, and superimposes our definitions.

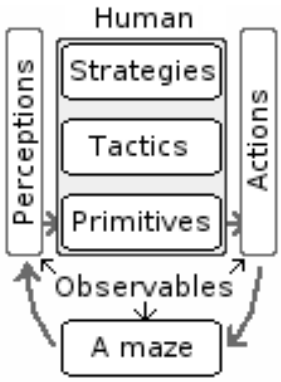

Figure 3. From observables to strategies

The raw data contained in the databases, called the observables, are indicated in fig. 2. They are the basic facts such as the position of the person in the maze at a given time step, the position of his/her hands, etc. 50 observables were recorded every quarter of a second in what we call the "log files" or "databases". Each maze also has a static description indicating the position of the obstacles, of the treasure, etc. Primitives are combinations of observables, and sometimes of observables and static maze descriptors. We call "tactic" a combination of observables and primitives linked by a common goal or sub-goal. We defined four tactics:

- the goal-related treasure hunting tactic, called the "search tactic",
- the tactic used by the volunteer to cope with the fact that he or she has to move around blindfolded, called the "moving tactic",
- the tactic causing the behaviour of the volunteer encountering an obstacle, which has a mixed purpose of treasure hunting and spatial orientation, called the "obstacle following tactic"
- the personal safety tactic called the "obstacle detection tactic".

In [11] we detail why and how this particular decomposition was chosen. The combined effect of enacting each of the four types of tactics is a strategy.

Fig. 4 shows the path we followed in this work: First a bottom-up generalisation, in several steps, which started with the log file recording the movements of the subject and was achieved with the help of machine learning algorithms. Then the top-down implementation of

[6] Our replicates are only "close" because the recordings are noisy. We believe this noise contributes to the robustness of the ensuing generalisations but we did not test this hypothesis.

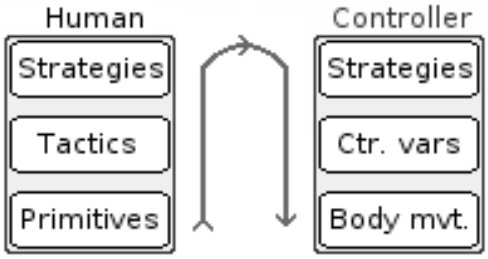

Figure 4. From primitives to body movements

the induced strategies into control variables ("Ctr. vars" in fig. 4) and robotic body movements ("Body mvt").

6 A formal definition of "strategies"

A situation variable is a descriptor of perceptions of the environment external to the controller[7]. Each of the M situation variables has a finite and known number of possible values.

A control variable is a descriptor of robot action. Each of the N control variables has a finite and known number of possible values.

Formally, a robotic strategy is:

- A finite set of external situation states, E. Each situation state of E is expressed by a vector of M situation variables values: (e_1, ..., e_M).
- A finite set of internal action states, I. Each action state of I is expressed by a vector of N control variables values: (i_1, ..., i_N).
- An action transition matrix mapping all possible situation states to all possible action states. The values contained in this matrix are the probabilities of the robot enacting the behaviour described by an action state given a situation. We call it $\Lambda_A = a_{ij}$.
- An action duration mean transition matrix mapping all possible situation states to all possible action states. The values contained in this matrix are the means, should the robot enact the behaviour described by an action state, of the duration of all control variables of this behaviour. We call it $\Lambda_{AD} = a_{d-ij}$.
- An action duration standard deviation transition matrix mapping all possible situation states to all possible action states. The values contained in this matrix are the standard deviations, should the robot enact the behaviour described by an action state, of the duration of all control variables of this behaviour. We call it $\Lambda_{ASD} = a_{sd-ij}$.

Whenever the situation state of the robot changes, the robot goes into a certain action state chosen randomly according to the probabilities of Λ_A. It draws durations, in independent draws, for all the control variables values according to the Gaussian probability distributions defined by Λ_{AD} and Λ_{ASD} and starts a countdown to implement these durations.

When the situation state of the robot does not change but one of the control variable values reaches the end of its randomly assigned number of time steps, the robot goes into another action state chosen randomly, according to the probabilities of Λ_A, among all action states which have the same values for all the other control variables and a different value for the control variable which is due for a change. It draws a duration for this new control variable value according to the probability distributions defined by Λ_{AD} and Λ_{ASD}.

[7] Not necessarily "external to the robot", the input from a sensor describing the state of the internal battery would be a situation variable value.

A discussion of this model and of how it compares to Hidden Markov Models can be found in [12], and there we explain that brute force learning of such a model would be intractable, not least of all because it would require enormous log files. We were not the first to say that such transition matrices are intractable, it has been known since the eighties, this is why we use several steps of generalisation and constrain our matrix. We cannot take one giant step from perception to cognition but we can climb this hill by taking several consecutive small steps, each time building upon the result of the previous generalisation step, and looking ahead to provide guidance to our learning algorithms.

7 Changes of strategies

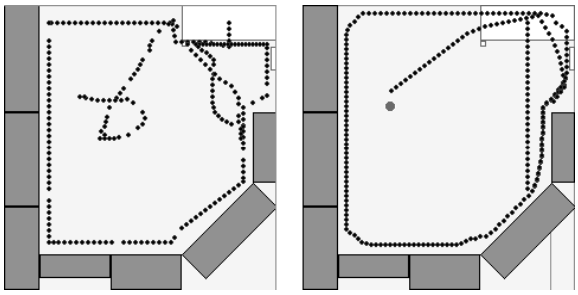

Figure 5. Track of G7_2's run (left) and of our robot implementing her strategy (right)

How do humans recognise failure? In [12], we document an example of a volunteer, G7_2, changing her search strategy after coming across the same unique object for the second time: she apparently realised she was going round in a circle (fig. 5). Testing for possible failures can be part of a global search strategy, again not one which could be learnt in one step from reasonably-sized logs but one which requires a multi-levels control architecture and step-by-step learning. G7_2's behaviour was learnt because from the observables "right hand (X, Y) coordinates", "left hand (X, Y) coordinates" and "unique object number N (X, Y) coordinates" the primitives "RHU" and "LHU", right or left hand in contact with a unique object, were built[8]. During our statistical analysis (documented in [11]) the connection between coming in contact for the second time with a unique object and the "going off into empty space" behaviour, primitive of the "moving tactic", became apparent. Failure recognition can lead to a change of strategy, which consecutive enaction can be observed.

In our formalism, switches from one action state to another can be triggered both externally, by a change of situation state, and internally, by the reaching the end of some control variable value life span. When the log file shows such a switch happening independently of these two conditions, it corresponds to a change of strategy. Changes of strategy are defined by a subset of situation states, either of which triggers the change, by a consecutive sequence of situation states belonging to this subset, or by a time limit assigned to each consecutive strategy (when they are programmed by hand).

[8] The observables are outside the controller, they are the raw data observed on the videos. The controller has no access to (X, Y) coordinates, it only has access to the knowledge that the right or the left hand is in contact with a unique object and encounters with a unique object are numbered.

We are interested in how robots can learn problem-solving strategies by human imitation. We are not interested in the strategies themselves. We deliberately make no selection related to the efficiency of the strategies our robot imitates. Our system observes subjects, analyses and implements not only winning strategies but all observable strategies.

Human beings have several "general purpose" problem solving strategies at their disposal [1]:

- Means - end analysis
- Working forwards / working backwards
- Induction
- Planning
- Trial and error
- Divide and conquer
- Etc...

We are interested in all of them, not only in winning strategies, because even if a given strategy, for example trial and error, may not be very efficient in our context, it is efficient in other contexts and the fact that it can be learnt by imitation is in itself important. And failures are important. The best-known example is probably the error backpropagation algorithm [8]. It is applied by hypothesis generate-and-test learning methods of all kinds. Learnt and/or evolved robot controllers with predictive capabilities test their predictions, sometimes at each time step, sometimes later on, sometimes only at the end of the run. Whole control architectures can be made to evolve when they fail some test.

The tests which failure drives learning and/or evolution range from the very specific hypothesis "at the next time step sensor 18 will have a value of 12" to the global "the goal will be reached". In the mazes experiments, the volunteers resort to the global test, but also make up their own testable sub-goals as they go along. For example, a sub-goal can be to ascertain that the treasure is not on a given table before moving on to the next table. The failure of the corresponding test leads to backtracking and exploring the table again.

8 The H-CogAff architecture, the global alarm system and bumps

One of the more promising approaches to understanding what constitutes a mind seems to be modular multi-layered approaches such as H-CogAff.

Modularity enables divide-and-conquer problem-solving. So H-CogAff is implementable, albeit for the moment only in a very limited and schematic way, and can thus be tested.

Our tests showed us that because of its multi layers H-CogAff supports learning human problem-solving stategies by imitation, a human capability. This multi-layers caracteristic enables step-by-step learning, solving an otherwise intractable problem.

The next set of tests, which this paper is about, showed us that adding a global alarm system (GAS) improved the performance of our simulated robot. The GAS does not only "make sense", we proved that it actually works as predicted in our limited implementation of the H-CogAff model.

In the rest of this section we discuss parts of the H-CogAff, and in particular the GAS, comparing the model with our implementation of it. Testability being important for validating this model, we also discuss some possible observable effects that the GAS could have upon human beings. Finally we explain how we decided to test it and justify our decision by presenting the available data.

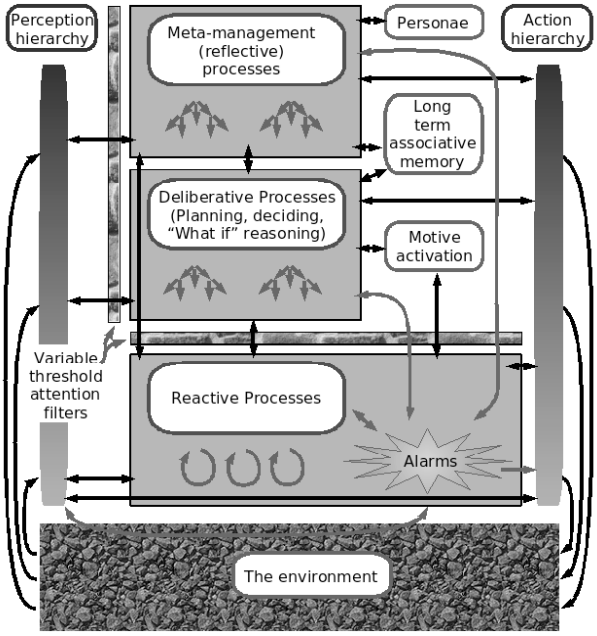

Figure 6. One version of H-CogAff

Fig. 6 is a more detailed representation of one version of the H-CogAff [21] architecture than the simplified one we showed in fig. 3. It has many similarities with robot control architectures (see for example [25] and [16] but many more exist which are just as interesting). In our previous model we implemented the environment, input, output and the three layers. Our statistical inference could be seen as corresponding to the long term associative memory. We absolutely do not claim to have implemented modules which even approach the complexity of the corresponding processes in the human brain. Our current model was built focusing upon the detection of changes of strategy, and H-CogAff was our inspiration.

According to H-CogAff, the human mind operates on three levels, each with its own functionality in terms of processing information. The lowest module senses the outside world, taking in stimuli from its infinite variety of sources, senses the inner state of the body and takes care of muscular control. It holds a summary of its perceptions and actions, written in a higher-level language, at the disposal of the module above. It has both a short term and a long term memory. The short-term memory is illustrated by play-back capacities: if we move our hand without being aware of it, maybe because we are shifting position, and put our hand down on something squishy, this both triggers the alarm system which draws our attention to the fact and offers us a playback of the movement we just did. Even if we automatically withdrew our hand after the event this playback enables us to know where it had been. The long term memory can store, probably in the very shape of the processes taking place in this module, the result of learning propagated down from higher modules. In natural language processing this lower module learns to do its equivalent of speech-to-text and text-to-speech. When we learn to drive we have to think about shifting gears but as we become more experienced this whole sequence of finely tuned muscular movements drops below the attention threshold. We drive on, secure in the knowledge that if something went wrong, if the gear level jammed, our lower module would immediately report failure via the alarm system. Without the learning abilities of the lower module even very day-to-day actions

such as brushing our teeth would become overwhelmingly complex.

In our system, learning was also required to define for example the type of trajectories the volunteers followed when walking across empty space. We could set the "class" variable[9] to 2 when the volunteer was walking approximately straight, to 1 when he or she was walking along an approximate curve and to 0 when he or she was approximately U-turning. Then the inductive algorithm[10] combined the values of differences in position and direction between current and shortly passed time steps and found good descriptions of these primitives.

The global alarm system is a mechanism enabling the lower-level processes, which for most of the time take place below the attention threshold, to suddenly draw attention to themselves when something goes wrong. A person can walk without thinking about it but if this person stumbles he or she will immediately pay attention to his or her feet and their immediate surroundings. The power of the global alarm system is such that it can "take over" the whole system, switching action control from the upper levels down to its own. If it detects imminent physical danger it can trigger a release of adrenaline, and people having experienced this report that afterwards they need a noticeable lapse of time before they can get their higher-level thought processes back to normal. The global alarm system can also send messages straight to the emotion centre (not a specific module in H-CogAff), for example when we are having a conversation, and paying attention to what is being said, it can suddenly report "the person you are talking to is very attractive", with potentially disruptive consequences as far as the conversation is concerned. So the global alarm system can trigger a change of strategy, in this case the strategy changes because the goal changes, from "how can I convince this person that my point of view is better?" it becomes "how can I convince this person to have dinner with me?". So an imitation system whose purpose would be to emulate this subject's behaviour would have to learn two distinct strategies, the one which took place before the alarm and the one which took place after it. Detecting changes of strategy, in our case abrupt variations in the primitives values distributions, cannot be used to teach a robot's global alarm system the set of circumstances in which it should send out an alarm until we have a way of differentiating the changes triggered by the alarm system from the changes triggered by some other process. But, given a set of alarm signals and the time step at which they occured, we can test whether it makes sense to consider that the strategies taking place before and after the alarm are different, and this is what we did.

When the videos of the psychological experiments were converted to log files, variables were recorded which were later discarded both by our attribute construction algorithm and by us, for example the direction in which the head was facing. The purpose of the algorithm at that point was to construct primitives from observables and it had no look-ahead capacities which would have enabled it to estimate the usefulness of the observables upon the detection of changes of strategies: in our step-by-step approach the inductive algorithm was looking at the next module up and not right to the top.

One of the discarded observables was the binary "bump" variable. The "bump" observable took a value of 1 when the volunteer bumped into an obstacle or a wall and it's value was 0 otherwise[11]. Among our 50 observable variables, "bump" was the one which seemed to have the highest likelihood of triggering the global alarm system of our volunteers. So we decided to set "bump" as a situation state which forced a change of strategy.

[9] In machine learning, the "class" variable is an exogenous variable which value specifies to which category the example belongs

[10] We tried a few and C4.5 [17] and [18] turned out to be the most efficient for our purpose

[11] We had many descriptors for the "obstacle following" behaviour but none in any of the higher modules which took into account whether the volunteer had come into contact with the obstacle gently or whether he or she had bumped into it.

9 The variables which values control robot actions

These variables, called "control variables", express by their values what the robot does. Our 6 control variables are the basic bricks from which the human-like strategies were implemented. We had 36 primitives built from 50 observables, and 4 tactics. We explain why there are now 6 control variables. The robot controller programming language that we implemented gives the actions which should be performed in all possible situations. One value of a control variable corresponds to one action of one part of the robot's body. Our programming language expresses, in probabilistic terms, when a control variable value should change and to which new value it should change.

When we reached this stage we knew how to express the strategies in terms of tactics and the tactics in terms of primitives and observables. So we sorted our task-relevant actions according to body parts and possible contexts. The legs are considered a single body part. This was also a way of eliminating the conflicts which would certainly have arised, in a probabilistic context, if we had attempted to implement the tactics independently one from the other.

The control variable are in the middle level, the lower level of the controller, "body movements", does not need to know whether the robot should move forwards because it is part of its search tactic or because it is part of its obstacle following tactic.

9.1 EXPLORE_OB: Object exploration behaviour

Exploring an object means in this context feeling its surface with one or two hands. As the robot goes along an obstacle, the hand on the side towards the obstacle can either follow the edge of this obstacle, sweep the obstacle, or stay by the body side. This control variable defines what the hands are doing relatively to an obstacle when the robot is near an obstacle. Sweeping a wall, apparently in search of top shelves, was rarely done by the volunteers and was in contradiction with the information the volunteers had been given, namely that the treasure would be on the ground or on a table but not on the ground under a table. So we did not implement this behaviour.

EXPLORE_OB has five possible values:

1. None: The hands stay by the body side or searches the empty space in the obstacle detection behaviour.
2. 1HnE: One hand not efficient. One hand follows the near side of the object, only covering a small percentage of its surface as the robot goes along. The other one is a the body side or exploring the empty space.
3. 2HnE: Two hands not efficient. Both hands act as described above. They are following one another along the near edge of the obstacle the robot is following.
4. 1HE: One hand efficient. One hand sweeps the whole surface or nearly the whole surface as the robot goes along.
5. 2HE: Two hands, at least one acting efficiently.

9.2 EXPLORE_GR: Ground exploration behaviour

There are three ways the robot can explore the ground. They can occur near an obstacle or in empty space, and can coincide with strictly

positive values of the obstacle exploration variable, though when the ground exploration variable is equal to 4 the obstacle exploration variable can only be equal to 1 or 2.

EXPLORE_GR has four possible values:

1. None: No exploration of the ground
2. Walk: The simplest, and least efficient ground exploration, is walking.
3. Sweep: The robot can "sweep" the ground, emulating the behaviour which consisted for a person of swinging his/her legs at each step to cover more ground than by walking.
4. Bend: The robot can emulate the behaviour of a person bending down or squatting and exploring the ground with his/her arms. This is the most efficient ground exploration technique.

9.3 EMPTY_SPACE: Empty space behaviour

The volunteers in the mazes, when they ventured into empty space, strongly tended to either go approximately straight or to U-turn along a half circle of narrow radius. This control variable describes the trajectory in empty space.

EMPTY_SPACE has four possible values:

1. Not_Walk: Not walking across empty space.
2. Straight: Walking approximately straight across an empty space.
3. Curve: Walking along a approximate curve of large radius.
4. Turn: Walking along a approximate curve of narrow radius.

9.4 GO_OFF: Going off into empty space behaviour

This variable controls what happens when the robot is near an obstacle in terms of will it stay near it or will it go away into empty space. Going from one obstacle to another, for example when following a row of tables, is not going off into empty space.

GO_OFF has two possible values:

1. No: The robot does not go away from the obstacle
2. Yes: The robot chooses a random open direction and takes a step in that direction. It can go straight away from the obstacle or nearly tangentially to it, getting only marginally farther from it during the first steps.

9.5 FOLLOW_OB: Obstacle following behaviour

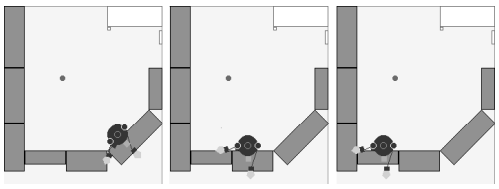

Figure 7. A time sequence of screenshots of a simulated robot enacting EXPLORE_OB = 2HE and FOLLOW_OB = Serious

The "Careless" vs. "Serious" qualifications for the obstacle following variable have to be understood as applying specifically to the obstacle following behaviour. Some people in the maze were seen to be sometimes following an obstacle, or a row of obstacles, quickly and efficiently. This could happen when they had just met a known unique object and were backtracking away from it. They were then only interested in the obstacle as a guideline. Fig. 7 illustrates an obstacle following behaviour and an object exploration behaviour in the G7 maze. These screenshots do not illustrate consecutive time steps because our time steps are 0.25 second appart and the difference not so easy to see.

FOLLOW_OB has three possible values:

1. Not_Follow
2. Careless
3. Serious

When an obstacle is encountered, if the going off variable has a value of "No" and the obstacle following variable has a value of "Not_Follow", the robot will just stay where it is.

9.6 OB_DETECTION: Obstacle detection behaviour

Figure 8. A volunteer whose actions would be expressed for the robot as EXPLORE_OB = 1HnE and OB_DETECTION = 1_Hand

The blindfolded volunteers in the mazes often slowly waved their hands in the air, with their arms held horizontally or with their hands held approximately at the height of the last obstacle encountered. One example of this behaviour is shown in fig. 8. This behaviour is what the obstacle detection variable describes.

OB_DETECTION has four possible values:

1. None
2. 1_Hand: When the person is following an obstacle and the obstacle exploration variable indicates that this obstacle is only being explored with one hand, it is still possible for the robot to do obstacle detection with the other hand.
3. 2_Hands: This can only happen when the person goes off or is in empty space.
4. Bump: "Bump" was implemented for completeness as a robot control variable possible option: we wanted all the observables to be reproducible, but the result of this control variable being set to "bump" was simply to make the robot rush straight into the first obstacle that happened to be in front of it and it was never included in any of our automatically generated robot control programs.

9.7 Implementation of human-like behaviour

A human-like behaviour is one which does not deviate by more than 3 standard deviations from the average of all recorded human behaviours (as shown in [11]). The control variables values are set according to average duration and not average value. This is illustrated by fig. 9: If the binary variable represents "Moving forwards", the top part of fig. 9 would be a slow walk, the discretisation in time steps and integer distance units filling the log with alternating values 1 and

0. The lower part would be a fast walk followed by a pause. The average value of this variable is obviously the same top and bottom, one half, so these two very different behaviours can only be distinguished by the average durations of consecutive series of values.

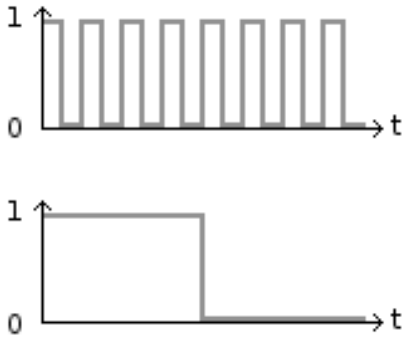

Figure 9. Same average, different behaviour

Learning was necessary during the bottom-up analysis, but, once the model was built, finding the average durations and their standard deviations to implement a given strategy only requires counting, given a specific context, the various corresponding occurrences in the log file. For example in the empty space context a volunteer could walk straight ahead for 12 time steps on average with a standard deviation of 3 and along a curving trajectory for 10 time steps on average with a standard deviation of 5. The values implementing this behaviour were randomly drawn with the corresponding probability distribution.

Without attempting to implement changes of strategies within a run of a volunteer in a maze the most efficient program was the one which considered all the 10 strategies together, independently of the duration of the run [12].

To implement our very limited version of the H-CogAff global alarm system, we set our program to divide the log files into several segments according to the bumps. As can be seen in table 1 the runs of 4 volunteers were not segmented because these persons never bumped into an obstacle. So these runs are considered to be driven by a single strategy all along. Twice, two bumps occured shortly one after the other so the number of values between them was insufficient for a significant description of a strategy and the second bump was ignored. We ended up with 27 strategies which we combined.

Table 1. Durations and numbers of bumps

Run	G1_1	G1_2	G1_3	G3_1	G3_2
Dur.	1'50"	31'48"	12'53"	1'33"	1'44"
Bumps	0	4	4	0	0
Run	G4_1	G4_2	G4_3	G7_1	G7_2
Dur.	18'46"	18'37"	18'29"	4'54"	4'43"
Bumps	6	1	2	2	0

Next we describe how we combined strategies in this case. If all the combined strategies are given equal importance, the length of the corresponding run notwithstanding, the result of dividing a run according to bump occurrences increases the relative importance of the strategies of the volunteers who bumped into obstacles more than others. In this case any improvement could be shrugged off with considerations such as a more active or even a more daring attitude is beneficent to search efficiency as well as being detrimental to personal safety. So on one hand we combined all strategies occuring during one minute after a bump together into the "after bump" strategy, and on the other hand we combined the strategies taken from the rest of the runs into the "no bump" strategy. The one minute limit was chosen arbitrarily. When the robot bumped into an obstacle it abruptly switched to the "after bump" strategy, all probabilistic values being redrawn at once. One minute later the robot started implementing the "no bump" strategy but this time the change occured more gradually, each variable finishing the life span assigned to its current value before drawing a new value according to the "no bump" strategy.

10 Experimental results

We consider that the fact that some human problem-solving strategies are learnable by a robot is more important than the actual strategies being learnt. It must also be remembered that the experimental settings were strongly constrained and that using bumps as the sole trigger of our "global alarm system" does not do justice to this complex system.

We call the result of the program generated from all logs indiscriminately joined together the result of robot_1 and the result of the program generated from the two distinct sets "after bump" and "no bump" the result of robot_2.

10.1 Reasons for the lack of comparisons between robot and subject performance

The task of the volunteers was to find the treasure (or, in some cases, for one of them to find the treasure, after which they could all walk out). So their performance in terms of time to achievement and/or ground coverage depended too much upon the original location of the treasure for performance comparisons to be made. We only had the logs of ten subject runs. Each subject only went through a maze once. They did not all go through the same maze. Three subjects definitely did not spend all their time in the maze searching for the treasure. So such comparisons would in any case have had low statistical significance.

The logs of successful volunteers who found the treasure quickly, do not generate better controllers. In fact in one case it was quite the opposite: G1_1 used a systematic right-handed blind man walk: she followed the wall on her right, thoroughly exploring all objects she encountered on her right and ignoring anything on her left. She quickly found the treasure because it was located on the ground, near a radiator which was fixed to the wall. The corresponding controller makes the robot go round in circles following the outside perimeter of the maze and never explores the interior so it is not a good sweeper. G1_1 only covered about a tenth of the maze before finding the treasure. In terms of time to achievement she was fast, in terms of ground coverage she was very slow.

On average the subjects took 11'32" to find (or for one of them to find) the treasure. Robot_1 reached between 78% and 94% of all reachable squares at least once after 10 minutes and robot_2 reached between 80% and 94% of all reachable squares at least once after 10 minutes. Concluding that, given uniformly random treasure locations, the robots did better than the subjects would be, to say the least, premature.

Given these facts and lack of a lot more of human-related data we regretfully forwent subject-to-robot comparison measurements.

10.2 Robot results

In the following scale and speed correspond to the real settings.

The settings for the following were the four mazes from which one or several logs had been drawn, and six extra (invented) mazes used for testing purposes.

- The performances were not better in the "known" mazes than in the invented mazes, showing that the strategies had really been abstracted from their original settings.
- All tables had been explored after at most 11 minutes by robot_1. After 10 minutes robot_2 had explored all tables in all but one maze. That last maze had an isolated central table which robot_2 often bypassed in its exploration of the empty space.
- On average, 83% of the tables had been explored after 3 minutes by robot_1 and 86% by robot_2.
- Dividing the ground in squares 20 pixels across[12], which corresponds to the average "width" of a subject as seen on the videos, between 78% and 94% of all reachable squares (ground and tables) had been reached at least once after 10 minutes by robot_1, and between 80% and 94% by robot_2, the actual average values varying according to maze size and complexity. The improvement mostly happened in complex mazes with many tables.
- These percentages increase with run duration.

The difference between the efficiency of robot_1 and of robot_2 might seem slight, but as performance improves each percent point becomes harder to gain than the previous one. So the difference between 83% and 86% of the tables is more significant than would have been a difference between 63% and 66% of the tables.

11 Conclusion

Closing the loop: As our robots move about in the mazes, their observable actions can be recorded. Thus the teacher/learner/teacher loop, sometimes also refered to as the raw-data/trained-system/raw data (or raw-data/learned-strategies/raw-data) loop, is closed because new logs can be generated and learning can be achieved from these new logs. These new logs are neither a better nor a worse model than the originals, were the robot generating them to run for a sufficiently long time and in a sufficiently complex maze for the randomness to be overcome by statistical significance they would amount to the same information.

H-CogAff validation: H-CogAff describes the functions of the mind [22]. Our experiments only validate the H-CogAff model in a severely limited and constrained context. They also validate the global alarm system, with the same restrictions and more because a single variable was used to trigger strategy changes, and this variable was neither learnt nor even statistically determined but was chosen by empirical reasoning. We can however say that an instant reaction which resets every control variable upon a bump improves performance, and that this seems to indicate that simulating a global alarm system, hopefully in more advanced ways, in other robotic controllers could be a promising line of research.

ACKNOWLEDGEMENTS

Our link between the global alarm system and emotions was first suggested by David A. Focil.

[12] We were given the plans of the mazes by the psychologists, but these plans had no scale. When we inquired about it we were told that the corresponding rooms still existed and could be measured. We declined and used pixels for our distance unit.

REFERENCES

[1] Jean-Francois Richard Aldo Zanga and Charles Tijus, 'Implicit learning in rule induction and problem solving', *Thinking and Reasoning*, **10 (1)**, 55–83, (2004).
[2] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, 'Learning to imitate corresponding actions across dissimilar embodiments', *IEEE Transactions on Systems, Man, and Cybernetics*, **32**, 482–496, (2002).
[3] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, 'Correspondence mapping induced state and action metrics for robotic imitation', *Special Issue on Robot Learning by Observation, Demonstration and Imitation*, (2006).
[4] Billard and Hayes, 'Drama, a connectionist architecture for control and learning in autonomous robots', *Adaptive Behaviour*, **7**, 35–64, (1999).
[5] A. Billard and R. Siegwart, 'Robot learning from demonstration', *Robotics and Autonomous Systems*, **47**, 65–67, (2004).
[6] S. Calinon and A. Billard, *Imitation and Social Learning in robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, Cambridge University Press, 2007.
[7] S. Calinon, F. Guenter, and A. Billard, 'On learning, representing and generalising a task in a humanoid robot', *IEEE Transactions on Systems, Man and Cybernetics, Part B, Special Issue on Robot Learning by Observation, Demonstration and Imitation*, **37 (2)**, (2007).
[8] G. E. Hinton D. E. Rumelhart and R. J. Williams, 'Learning internal representations by error propagation', *David E. Rumelhart and James A. McClelland*, **1**, (1986).
[9] Demiris and Hayes, *Imitation in animals and artifacts*, 2001.
[10] R. Dillmann, 'Teaching and learning of robot tasks via observation of human performance', *Robotics and Autonomous Systems*, **47**, 109–116, (2004).
[11] M. Felkin, 'Learning by observation and induction the strategies of humans placed in a problem-solving context', *PhD Thesis*, (2008).
[12] M. Felkin and Y. Kodratoff, 'High level imitation learning in a treasure hunting task', *Proceedings of Plan, Activity, and Intent Recognition (PAIR) workshop, International Joint Conferences on Artificial Intelligence (IJCAI 2009)*, (2009).
[13] Valentina Iemmi, 'Les apports des nouvelles technologies a la psychologie clinique: Les robots comme compagnons therapeutiques', (2005).
[14] Heumer G. Jung B., Ben Amor H. and Weber M., 'From motion capture to action capture: A review of imitation learning techniques and their application to vr-based character animation', *Proceedings ACM VRST*, (2006).
[15] Henri Kraus, 'A formal theory of plan recognition and its implementation', *Reasoning About Plans, by J.F. Allen, H.A. Kautz, R.N. Pelavin, and J.D. Tennenberg*, 69–126, (1991).
[16] J. P. Müller, 'A conceptual model of agent interaction', *Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94)*, 389–404, (1994).
[17] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.
[18] J. R. Quinlan, 'Improved use of continuous attributes in c4.5', *Journal of Artificial Intelligence Research*, **4**, 77–90, (1996).
[19] S. Schaal, A. Ijspeert, and A. Billard, 'Computational approaches to motor learning by imitation', *Philosophical Transactions of the Royal Society of London: series B, Biological Sciences*, **358**, 537–547, (2003).
[20] Storz J. Shon A. and Rao R., 'Towards a real-time bayesian imitation system for a humanoid robot', *Proceedings of the IEEE International Conference on robotics and Automation*, 2847–2852, (2007).
[21] A. Sloman, 'Varieties of affect and the cogaff architecture schema', *Proceedings Symposium on Emotion, Cognition, and Affective Computing AISB'01 Convention*, 39–48, (2001).
[22] Aaron Sloman, 'Some requirements for human-like robots: Why the recent over-emphasis on embodiment has held up progress', *Creating Brain-Like Intelligence, LNAI 5436*, 248–277, (2009).
[23] C. Tijus, N. Bredeche, Y. Kodratoff, M. Felkin, C. Hartland, V. Besson, and E. Zietti, 'Human heuristics for a team of mobile robots', *Proceedings of the 5th International Conference on Research, Innovation and Vision for the Future (RIVF'07)*, (2007).
[24] Mark P. Woodward and Robert J. Wood, 'Using bayesian inference to learn high-level tasks from a human teacher', *International Conference on Artificial Intelligence and Pattern Recognition (AIPR-09)*, (2009).
[25] Victor Lesser Xiaoqin Zhang and Tom Wagner, 'A layered approach to complex negotiations', **2(2)**, 91–104, (2004).