Imitation With ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments

Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn

***Abstract*—Imitation is a powerful mechanism whereby knowledge may be transferred between agents (both biological and artificial). Key problems on the topic of imitation have emerged in various areas close to artificial intelligence, including the cognitive and social sciences, animal behavior, robotics, human–computer interaction, embodied intelligence, software engineering, programming by example and machine learning. Artificial systems used to study imitation can both test models of imitation derived from observational or neurobiological data on imitation in animals and then apply them to different kinds of nonbiological systems ranging from robots to software agents. A crucial problem in imitation is the *correspondence problem*, mapping action sequences of the demonstrator and the imitator agent. This problem becomes particularly obvious when the two agents do not share the same embodiment and affordances. This paper describes a new general imitation mechanism called Action Learning for Imitation via Correspondence between embodiments (ALICE) that specifically addresses the correspondence problem. The mechanism is implemented and its efficacy illustrated on the "chessworld" testbed that was created to study imitation from an agent-based perspective, i.e., by a particular agent in a particular environment.**

***Index Terms*—Correspondence problem, embodiment, imitation, machine learning.**

I. Introduction

A CHARACTERISTIC of many social animal species, e.g., dolphins, chimpanzees, humans, and other apes, is the ability to learn from others by imitation (see [30], [31], [39], and [53]). Inspired by nature, over the past decade, many researchers have attempted to design life-like social agents, i.e., software or robotic artifacts that are able to learn from each other or from human beings by imitation [4], [8], [10], [19], [26], [27], [33], [34], [43]. On the one hand, a robot or a software program that a human can teach simply by showing or demonstrating what needs to be done is an exciting new programming paradigm [6], [17], [35]. On the other hand, imitation also plays a crucial part in the development of humans and some other animals as social beings. Robots or software systems that possess imitative skills are therefore an important step toward truly social artifacts (see [18], [19], and discussions in [20]).

Manuscript received January 30, 2002; revised July 23, 2002. This paper was recommended by Associate Editor B. J. Oommen.

The authors are with the Computer Science Department, Adaptive Systems Research Group, University of Hertfordshire, Hatfield, Hertfordshire AL10 9AB, U.K. (e-mail: a.alissandrakis@herts.ac.uk).

Digital Object Identifier 10.1109/TSMCA.2002.804820

In the area of social learning, an animal learns benefiting from the presence of or the experiences of another animal, and such influences may support adaptation. Biologists and psychologists who study social learning in animals usually want to know whether nonhuman animals can imitate behaviors they observe. This provides an opportunity to both investigate the cognitive abilities of animals and compare them to those of humans, and also to help understand the role of social interactions in the development of patterns of behavior.

Many theories of imitation are discussed in the literature. In the following, we discuss two particularly influential and relevant examples.

Byrne proposes *string parsing* as a theory of imitation, which separates the copying of behavioral organization from an understanding of cause and effect in observed behaviors [14]. The recurring patterns in the visible stream of behavior (composed of structurally ambiguous strings) are detected and then used to build a statistical sketch of the underlying hierarchical structure, which in turn may aid to comprehend cause and effect. The linear sequence elements might represent simple behaviors recognizable by the animal or discriminably different states of the physical world affected by the behavior. His theory, however interesting, is highly underspecified with respect to the actual underlying *mechanisms*. The work presented in this paper has been partly inspired by this theory and develops computational algorithms for imitation of observed behaviors.

Another promising theory of imitation that makes testable predictions is Heyes and Ray's Associative Sequence Learning (ASL) theory [29], [31]. This theory assumes that a sequence of action units composes the behavior to be imitated rather than it being unitary. The resolution of these action units can vary, depending on the observer's perception. In order for an observer to imitate a sequence, ASL requires two processes to successfully take place. The horizontal process associates the representations of these action units in a successive chain, so that observer knows what the sequence "looks like." The vertical process associates directly or indirectly each of the sensory representations of the action units to appropriate motor representations, so that the observer can know how to perform the sequence. In our work, the vertical associations are captured by a *correspondence library* (see Section IV), while the temporal sequence of demonstrator actions relates to the horizontal process of ASL. ASLs primary purpose is to stimulate the development of other testable models of imitation and guide analytic experiments. One of the strengths of ASL is that it can also be applied to relatively *perceptually opaque* actions, i.e., facial ex-

pressions, that yield dissimilar sensory inputs when observed and when executed. One of the weak points of ASL is that it does not address effects on the environment; however, our framework (see Section II) allows this to be handled using appropriate metrics.

Note that the above two theories are complementary: Heyes and Ray's ASL theory aims to detect and copy sequential ordering, whereas Byrne's string parsing aims to extract hierarchical organization or structure from sequential ordering, and thereby copy at higher hierarchical levels. In Byrne and Russon's terms [15], ASL deals with action-level imitation, whereas string parsing with program-level imitation.

Many robotics researchers are inspired from biology for creating controllers for autonomous robots. Imitation is a powerful learning tool when social interaction either between humans and robots or even in multirobot systems takes place. Having a robot observe and learn to perform a task from an experienced teacher presents a more flexible and adaptive solution than explicit programming of each behavior. The learning process can be faster as no direct teaching is required; the expert, by just performing and demonstrating the task, can pass the required knowledge to the robot, which in turn may be used as a demonstrator and imitated by other robots, as shown in [9]. Robotics research in imitation often separates the mechanism from the social dimension of imitation, developing architectures that (usually using a vision system) identify salient features in the movements of a model and map them to appropriate motor outputs of the robot imitator [33], [34]. Focusing on the question of *how* to imitate given a particular robotic system and a specified task leads to very diverse control approaches that are difficult to generalize across different platforms and contexts. An exception to this is the architecture Dynamical Recurrent Associative Memory Architecture (DRAMA) for *learning by imitation* that has been applied to different robot platforms and contexts and is described in [7] and [8]; see also studies with physical and simulated robots in [27].

Learning by imitation is the area of study that investigates how an agent can exploit imitation as a means of acquiring knowledge, having solved the problem of how to imitate [27]. In the scenario discussed in [10], a robot learns properties of its environment by following another robot around, imitating its trajectory on a hilly landscape. In [11], it is shown that providing a robot with the ability of imitating a teacher agent enhances its performance at learning the rudiments of a synthetic proto-language. Imitation here can either be the means of enhancing the learning capabilities of the agent, as in the two previous examples, or more commonly is used to share a common context and replicate the actions of an experienced teacher.

Recently, there has been increasing interest in the use of imitative learning for the control of humanoid robots [6], [12], [36], [49]. Besides the ergonomic benefits when functioning in environments designed for humans, a humanoid robot can inspire a sense of familiarity that improves the social dimension of the interaction, and having the ability to imitate, a humanoid robot can be more believable and useful in social situations.

Imitation is also increasingly studied in software systems. *Behavioral cloning* is a method by which human subcognitive skills can be captured and reproduced in a computer program. As the human subject performs the skill, his or her actions are recorded along with the situation that gave rise to the action. A log of these records is used as input to a learning program. The learning program outputs a set of rules that reproduce the skilled behavior. This method can be used to construct automatic control systems for complex tasks for which classical control theory is inadequate. In [48], experiments are described where a flight simulator was modified to log the actions of human subjects as they were flying the aircraft. These logs were then used as input to an induction program that produced a decision tree from which autopilot code was derived.

A radically different approach to computer programming has developed since the early 1980s, called *programming by example* (PBE) [17]. It is also sometimes called *programming by demonstration* (PBD) because the user demonstrates examples of the desired behavior to the computer. The concept is that if the user knows how to perform a task on the computer, he or she should not need to learn a computer language in order to describe how to carry out the task. Presented with a number of examples, the computer should be able to derive a program corresponding to the user actions as they were performed and then be able to either repeat them or generalize the program to also work in similar situations [35]. Only recently have researchers in this area noticed the similarities with imitation studies, and there are only a few PBE systems inspired by cognitive frameworks so far. One of them, Learning Algorithms from Worked Examples (LAWE), was developed by Furse as a computer program that implements an imitation algorithm that operates according to some principles similar to the ones required for Byrne's string parsing theory [14], [24].

The actual *embodiment* of biological or artificial agents plays a critical role for intelligence, learning, and other issues studied in "Nouvelle AI" or "Embodied AI" (see [13] and [44]). Unlike other work on imitation that either provides an *ad hoc* mapping between demonstrator and imitator or assumes that they share identical embodiments, our work systematically investigates constructive solutions of the correspondence problem between dissimilar embodiments (see Section II).

The general framework that will be used for imitation across dissimilar embodiments is presented in Section II. The experimental testbed (the "chessworld") is described in Section III, while in Section IV, the generic imitating mechanism of ALICE is introduced. Experiments with ALICE implemented in the chessworld platform are presented in Section V and discussed in Section VI. Conclusions follow in Section VII.

II. General Framework

In order to study how to imitate, we must first define more precisely what we mean by imitation within the context of this work. A classical definition of imitation by Thorndike is "learning how to do an act from seeing it done" [51]; however, this definition is very open ended. We prefer Mitchell's more detailed definition, which allows for nonbiological agents and is necessary for imitation research in robotics/computer science. The following requirements are to be satisfied as evidence of imitation [38]:

i) Something C (copy) is produced by an organism and/or machine, where
ii) C is similiar to something else M (model);
iii) Regristration of M is necessary for the production of C;
iv) C is designed to be similiar to M.

Imitation is a powerful learning mechanism, and a more general *agent-based approach* must be used in order to identify the most interesting and significant problems, rather than the prevalent *ad hoc* approaches used in most imitation robotics research so far. Traditional approaches concentrate on finding an appropriate mechanism for imitation and developing a robot-control architecture that identifies salient features in the movements of an (often visually observed) model and maps them appropriately (via a built-in and usually static method) to motor outputs of the imitator (cf. one of the first examples of robotic imitation in [33] and [34]). Model and imitator are usually not interacting with each other, nor do they share and perceive a common context. Furthermore, the social dimension of imitation (and corresponding issues of when and why an agent should imitate) is usually ignored. Effectively, this kind of approach limits itself to answering the question of *how to imitate* for a particular robotic system and a particular imitation task. This has led to many diverse approaches to robot controllers for imitative learning that are difficult to generalize across different contexts and different robot platforms.

In contrast to the above, the agent-based approach for imitation considers the behavior of an autonomous agent in relation to its environment, including other autonomous agents. The mechanisms underlying imitation are not separated from the behavior-in-context, including the social and nonsocial environments, motivations, relationships among the agents, the agent's individual and learning history, etc. [21]. Such a perspective helps unfold the full potential of research on imitation and helps in identifying challenging and important research issues. The agent-based perspective has a broader view and includes five central questions in designing experiments on research on imitation: *who* to imitate, *when* to imitate, *what* to imitate, *how* to imitate, and how to *evaluate* a successful imitation. A systematic investigation of these research questions can show the full potential of imitation from an agent-based perspective.

A. Who to Imitate

It is important that the imitating agent chooses its demonstrator in such a way that engaging in an imitating behavior would benefit the imitator in some way. There is no need to imitate other agents whose tasks and needs are not relevant or beneficial. If the agent has to choose among several demonstrators, some evaluation of the performance of the appropriate behavior(s) by the possible candidate models is required before a choice is made. Note that the demonstrator is not required to be aware of the fact that the performed behavior is a model for the imitator, although this might help in providing feedback on the success of imitation.

B. When to Imitate

Once a suitable demonstrator is found, the imitating agent has to segment the entire demonstrator behavior, assigning to the behavior to be imitated a beginning and an end. The imitator also has to decide on a suitable time (and place) to imitate, i.e., whether a previously or currently observed behavior would be appropriate to carry out in the current context, and perhaps (if this is applicable) how many times to repeat.

C. What to Imitate

There are several aspects of a behavior that could be imitated. It may be preferable to imitate *states*, *actions*, or desirable *effects* of an observed behavior (or some combination of these) [16], [41], [42]. The structure of the knowledge transferred poses another problem, as there can be a distinction between different modes of imitation. Byrne and Russon propose two different kinds of imitation (*program level* and *action level*) as opposite ends of a spectrum, i.e., copying the organizational structure of the behavior versus copying the surface form of the behavior [15]. As a general consequence of this, an agent is required to have the ability to build hierarchical structures in order to exhibit program-level imitation.

D. How to Imitate and the Correspondence Problem

In addition to deciding who, when, and what to imitate, an agent must employ the appropriate mechanisms to learn and carry out the necessary imitating actions. The embodiment of the agent and its affordances will play a crucial role, as stated in the *correspondence problem*:

Given an observed behavior of the model, which from a given starting state leads the model through a sequence (or hierarchy) of subgoals in states, action, and/or effects, one must find and execute a sequence of actions using one's own (possibly dissimilar) *embodiment, which from a corresponding starting state, leads through corresponding subgoals—in corresponding states, actions, and/or effects, while possibly responding to corresponding events* [40]–[42].

This statement of the correspondence problem draws attention to the fact that the agents may not necessarily share the same morphology or may not have the same affordances even among members of the same "species." This is true for both biological agents (e.g., differences in height among humans) and artificial agents (e.g., differences in motor and actuator properties). Having similar embodiments and/or affordances is just a special case of the more general problem.

E. How to Evaluate the Imitation Attempt

For an attempt to imitate the demonstrator's behavior, there needs to be a measure to evaluate the behavioral matching. The choice of an appropriate metric is very important, as it will be used to capture the notion of the difference between performed and desired actions as well as the difference between attained and desired states [41], [42]. The evaluation might be performed either by the imitator, the demonstrator, or an external observer.

To show how the above issues come together for an imitating agent, let us consider the following example of a robot painter. If this robot observes several human workers in a house construction site, it makes sense to choose to imitate a human that paints the walls instead of a human electrician, whose work is not relevant to the role assigned to this robot. If there is a choice among

several painters, a brief evaluation of their individual work is required, based on criteria like how well they cover the wall surface, or whether they are messy and spill paint on the floor. This painting behavior should be imitated only if and when a wall needs painting. The robot painter can choose to imitate at action level replicating the exact same sequence of paint strokes on the wall as the demonstrator. Alternatively, it can choose to emulate the demonstrator, not necessarily using the same sequence or type of actions (for example, throwing a bucket of paint on the wall), but aiming for the same overall result of covering the wall evenly with paint. The robot must plan a sequence of actions and send appropriate control signals to its motors and actuators in order to move its limbs. The actual embodiment of the robot, its shape, the number and size of its limbs, together with the choice of a painting tool (a wider, larger brush will cover a greater surface but a smaller brush might be more precise), will play a crucial role, and a solution (even partial) to the correspondence problem must be found and used. Finally, as a possible evaluation measure, all the wall surfaces should have been covered, with no paint stains on the nearby windows or the floor carpet.

Research on imitation in robotics usually takes the approach of studying *learning by imitation*, assuming that an artifact already possesses the skill to imitate successfully and in turn exploits this ability as a means to acquire knowledge [10], [11], [27]. This paper addresses the complementary approach of *trying to imitate* or *learning how to imitate*, the study for specifying the necessary mechanisms by which observed and executed actions are matched, so that the agent can use imitation to learn how to perform useful behavior [18]. We investigate how different such attempts at imitation can be evaluated and quantified and illustrate possible mechanisms for solving the correspondence problem between demonstrator and imitator. Differences in embodiment between animals and robotic and software systems make it more difficult but not necessarily impossible to acquire corresponding behaviors.

III. Introducing Chessworld

In order to study the five main issues in imitation mentioned above, we introduce the generic testbed of *chessworld*, implemented using the Swarm multiagent simulation system. Preliminary results using this testbed were reported in [1] and [2]. The inspiration behind it comes from the need to create a shared environment for interacting agents of different embodiments and affordances. In the rules of the game of chess, each player controls an army of chess pieces consisting of a variety of different types with different movement rules (see Section III-D). We borrow the notion of having different types of chess pieces able to move according to different movement rules, and we treat them as agents with dissimilar embodiments moving on the familiar checkered board (see Fig. 1). The actual two-player game of chess is not studied, as it is not relevant to our work on imitation. Instead, we use pairs of dissimilarly embodied imitator and demonstrator agents to illustrate some interesting research problems in imitation, and we make use of the familiar context of chess in a generic way. Besides the familiarity, chessworld also benefits from having a simple discrete environment with well-defined and precise rules.

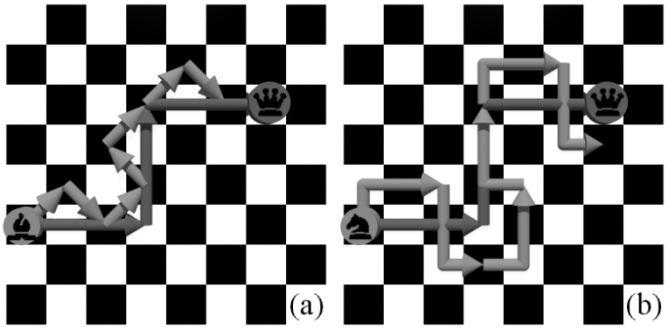


Fig. 1. Illustrating the effect of having dissimilar embodiments. The same demonstrator Queen (darker color moves) is imitated (lighter color moves) by (a) a Bishop and (b) a Knight.

The range of possible behaviors by the chess agents is limited to movement-related ones. As a demonstrator performs a random walk on the board, an imitator observes the sequence of moves used and the relevant displacement for each one of them and then tries to imitate this, starting from the same point. Considering the moves sequentially, the agent will try to match them, eventually performing a similar walk on the board. This imitative behavior is performed after the completion of the model behavior with no obstacles present, neither static (e.g., walls) nor dynamic (e.g., other chess pieces), besides the edges of the board which can obstruct movement.

A. Use of Different Granularities in Addressing What to Imitate

The data on the demonstrator's behavior can be structured and presented in a variety of ways, effectively reflecting different levels of complexity. More specifically, depending on the salience of the moves, important qualitative differences can be observed. We are going to discuss briefly three different levels of granularity of successively increasing levels of resolution:

1) *end-point level*;
2) *trajectory level*;
3) *path level*.

Considering only the starting and final locations, all the intermediate squares visited by the demonstrator will be ignored, and we will define this as using *end-point level granularity*. As a result, the trail of the agent as it tries to reach the overall destination can be qualitatively very different from the one shown by the demonstrator, although the behavior to be imitated is the same in respect to the overall result. The imitator emulates the overall goal (i.e., cumulative displacement) of the demonstrator (see Fig. 2).

If the imitator considers a list of locations to be reached sequentially, corresponding to the ones visited by successive moves of the demonstrator, the trail will be considerably more similar to the model one. We will define this as using *trajectory-level granularity*. In this case, as the effects of the individual movements (displacements) are considered sequentially (instead of in terms of their cumulative effect), a more detailed imitating behavior can emerge. This can be thought of

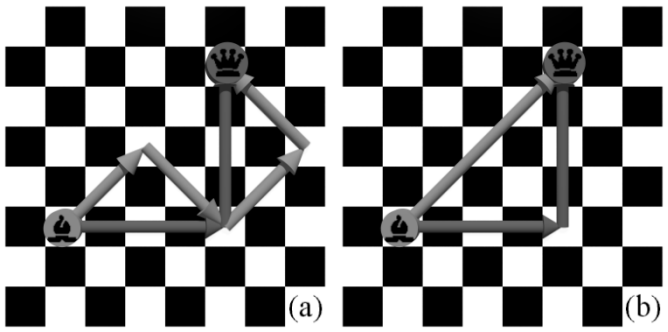


Fig. 2. Illustrating the effect of using different granularities. A Bishop (lighter color moves) imitates a demonstrator Queen (darker color moves) using (a) trajectory-level granularity and (b) end-point level granularity.

as program-level imitation with the task of reaching each of the sequential locations as subgoals.

In the game of chess, a piece can move to a new square as long as there are no obstacles in the way. In the current version of chessworld, no obstacles exist, nor do pieces take other pieces. We use the usual mental image of the pieces as sliding on the board, with the exception of the Knight jumping to a new location instantly, to define *path-level granularity*. More precisely, path-level granularity targets the list of locations, including not only the ones visited by the demonstrator as a result of its movements, but also the intermediate ones (if any). It represents a greater level of resolution than trajectory-level granularity. Path-level granularity can be thought as similar to action-level imitation, trying to replicate the trajectory of the demonstrated actions as closely as possible. The character of the resulting imitation depends on the granularity, and the success depends on how important the extra details are (see Section III-B).

In the current work, perception is very much simplified. The demonstrator's behavior is preprocessed and segmented according to the granularity to be used before being presented (noise-free) to the imitator agent. Generally, perception and the segmentation of actions are very important research issues in robotics, cognitive science, and computer vision, but the nature of the chessworld testbed allows us to abstract these away and to concentrate on issues specific to imitation.

B. Use of Different Metrics for Evaluating Imitation Attempts

In the previous section we described the possible resulting imitations in qualitative terms since we have not yet defined various notions of what a successful imitation is in the chessworld context. The agent behavior is composed of a sequence of moves and each move results in a displacement on the board, between the current and the previous location. Therefore a reasonable way to measure imitation success in this context is to measure the distance between the square the imitator agent moved to and the analogous square visited by the demonstrator, ideally that distance being zero for every salient displacement in demonstrated behavioral sequence. Other possible metrics (e.g., nongeometric, behavioral metrics) could be also used, including very complex ones [41].

There are various ways to measure the distance between two points on the plane, and for this work, three well-known metrics

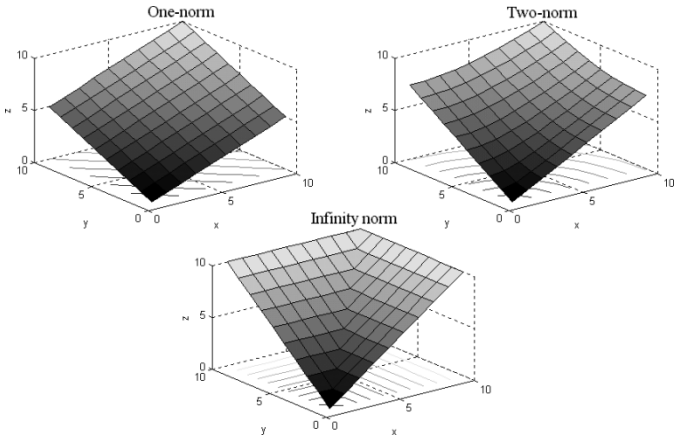


Fig. 3. Different metrics measuring the distance between current location $(0,\ 0)$ and subgoal location $(x,\ y)$. The vertical axes are normalized.

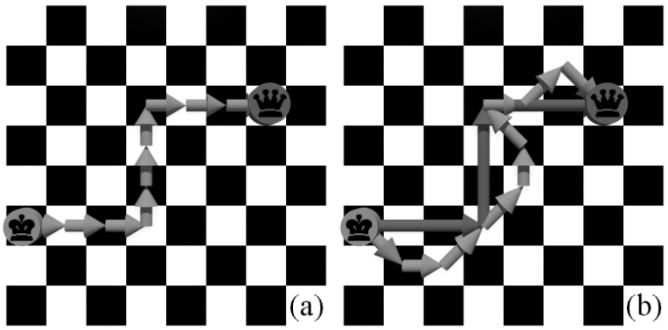


Fig. 4. Illustrating the qualitative effect of using different metrics. King (lighter color moves) imitates a demonstrator Queen (darker color moves) using (a) Euclidean distance norm and (b) infinity norm metric.

from mathematical analysis (e.g., [47]) were used. The distance between any two squares $(x_1,\ y_1)$ and $(x_2,\ y_2)$ on the chessboard is the following.

Hamming norm (or one-norm):

$$|x_1 - x_2| + |y_1 - y_2|$$

Euclidean distance (or two-norm):

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Infinity norm:

$$\max\{|x_1 - x_2|,\ |y_1 - y_2|\}.$$

The different metrics provide different notions of distance as visualized in Fig. 3. An example of the qaulitative effect of using different metrics is shown in Fig. 4.

In some cases, the imitator can get "blocked," i.e., unable to get closer to a subgoal by any single move, depending on the metric used. For example, the infinity norm value for a subgoal located diagonally to the current location to a Rook is the same value as the current location, and therefore, the metric cannot be further reduced.

Table I shows all possible cases in which such "getting stuck" can happen. Although this is caused because the default algorithm (see Section III-C) examines only *single moves*, rather than *sequences of moves*; nevertheless, it illustrates that

TABLE I
ALL POSSIBLE SITUATIONS, GIVEN BY A DESIRED DISPLACEMENT (x, y), WHERE NO POSSIBLE SINGLE MOVE CAN REDUCE THE VALUE OF THE METRIC. HERE, (x, y) IS THE DISPLACEMENT OF THE NEXT SUBGOAL RELATIVE TO THE CURRENT LOCATION

	Metric		
	Hamming	Euclidean	Infinity norm
Rook	---	---	x=y
Bishop	x=0 or y=0	(±1,0) or (0,±1)	(±1,0) or (0,±1)
Queen	---	---	---
King	---	---	---
Knight	(±1,0) or (0,±1)	(±1,0) or (0,±1)	(±1,0) or (0,±1) or (±1,±1)

the choice of metrics can be crucial for the evaluation and character of attempts at imitative behavior.

C. Default Imitation Algorithm

Given a sequence of demonstrator moves, the imitating agent has a list of locations on the board to be reached sequentially (depending on the chosen granularity), as well as a way to evaluate the imitation success for each of them (according to the chosen metric). To generate a sequence of moves that will define an appropriate imitating behavior, a simple algorithm with no elements of planning or learning can be used. The following lines of pseudocode effectively outline this default algorithm:

```
DEFAULT IMITATION ALGORITHM PSEUDOCODE
1. Observe demonstrator's moves.
2. Convert perceived moves to a sequence
   of subgoals for imitation depending on
   granularity.
3. For each subgoal displacement (x, y):
   3a. Choose a possible move that maxi-
   mally decreases the metric distance to
   (x, y).
   3b. Repeat step 3a until subgoal
   is reached or there is no move that
   strictly reduces further the distance
   to subgoal.
4. Move to next subgoal in sequence, if
   any, and repeat step 3, else stop.
```

D. Effects of Dissimilar Embodiment

If the demonstrator and imitator chess pieces are of the same type, then the process is very straightforward (for trajectory-level granularity). For each move of the demonstrator resulting in a displacement on the board, the imitator will use the same exact move to achieve the same displacement. The same repertoire of movements warrants very similar use of them, given a displacement to achieve, but in the case of dissimilar embodiment, more complex solutions can emerge with different degrees of success.

The movement rules for the chess pieces are an interesting combination. The *Rook* can in a single action, move an arbitrary number of squares but only horizontally or vertically, while the *Bishop* moves only diagonally. The *Queen* in a sense uses a superset of movement rules combining both their sets, while the *King* uses a related set that allows movement to all directions but only at one square at a time. The *Knight* has a very distinct style of moving, jumping to the opposite end of any 2×3 or 3×2 rectangle. The *Pawn* is omitted, since according to the rules of chess, it can only move a single square to only one direction, thus having significantly restricted movement capabilities for even simple imitation tasks.

We indicate compass direction and magnitude of individual moves notationally as follows: e.g., NW5, displacement of five units diagonally to the northwest or S1E2, displacement one unit south and two units east (by a Knight), etc.

Assigning the demonstrator and the imitator agents to be instances of different chess pieces creates a dissimilar embodiments scenario that requires solving the correspondence problem. The default algorithm will produce either a single action, or more probably a sequence of actions, corresponding to each action in the demonstrator sequence, while attempting to successively reduce the distance to the next subgoal (according to the metric used) at each step.

If moving to the target square cannot be achieved perfectly due to embodiment issues (i.e., the agent cannot move to that square due to its movement rules), the focus of the default algorithm will move to the next subgoal, having the current error added to the desired displacement. The demonstrator and the imitator have the same initial starting point, but for the rest of the imitation attempt, no correction takes place to put the imitator back on the trail, allowing these errors to accumulate.

For example, let us consider a Queen as the demonstrator that performs the action E3 (move three squares to the east). If the imitator is another Queen, the algorithm will simply produce the sequence [E3]. The same sequence will be produced if the imitator is a Rook. If the imitator is instead a King, the algorithm will produce the sequence [E, E, E] (three sequential moves of a single square to the east). If the imitator is a Bishop, the algorithm will produce [NE, SE] or [SE, NE]. Note that due to embodiment limitations (the Bishop cannot occupy the target square as it is of different color), moving according to either action sequence, the imitator cannot reach the desired square exactly, but only an adjacent one. Similar embodiment issues occur for an imitator Knight using the sequences [N1E2] or [S1E2].[1]

Although a corresponding imitation sequence is produced by the default algorithm and performed by the imitator, no learning and no association between demonstrator and imitator actions takes place at this stage. The most crucial limitation of using only this default algorithm to generate the imitating sequences is the lack of any planning. While every move in the produced sequence reduces the distance toward the current subgoal, a choice of a different move might be more beneficial in the long term, leading to a better location for subsequent subgoals but at the same time undesirably increasing the distance to the current subgoal. Such alternatives could potentially increase the imitation performance of the imitator agent. For example, the Knight, which cannot achieve a displacement to an adjacent square using

[1]To avoid confusion interpreting the action names, the entire Knight action set is {E1N2, E1S2, W1N2, W1S2, N1E2, N1W2, S1E2, S1W2}. We avoid multiple names for actions such as E2N1 and N1E2, which both would correspond to hopping two squares east and one square north, or, equivalently, one square north and two squares east.

this default algorithm (see Table I) can do so if it were allowed to temporarily increase its distance to the target.

IV. Introducing ALICE

Planning and learning can be added to the system by increasing the complexity of the algorithm, but such an approach might be more specific than generic, depending much on the problem domain and the context. To address this in an easy-to-generalize way, we introduce ALICE as a generic mechanism for building up correspondences based on *any* generating method for attempts at imitation by examining the history of such attempts (cf. Byrne's string parsing approach to imitation [14]). Preliminary results using the ALICE mechanism implemented in the chessworld were reported in [2].

Metaphorically speaking, the correspondence library that ALICE builds up functions as a kind of "refractive looking glass" or "prism" through which to transform a demonstrator's behavior into the repertoire of the imitator's own actions as constrained by its embodiment. Such a library of action correspondences can be employed when imitating (cf. the natural imitation of humans by dolphins [28] or robotic imitation [40], [41]). Mechanisms and correspondences of this type are also relevant to the imitation of perceptually opaque behaviors[2] and to sensory motor correspondences [31], to the extraction of the structure of demonstrated behavior [14], [52], and to neural mechanisms for the perception of actions and affordances and its direct mapping to motor actions via "mirror neurons" [5], [25], [46].

There are a variety of existing machine learning techniques addressing experience-based learning, for example, reinforcement learning [50], case-based reasoning [32], inductive learning [45], [48], or learning of behavioral histories [37], and others. Our intention here is not to develop a particular new and efficient machine-learning algorithm. Instead, we propose and systematically study a general framework for learning to imitate by solving the correspondence problem. Clearly, this framework could easily be combined with these or many other machine-learning techniques.

ALICE consists of two components on top of the arbitrary generating method used.

A. First Component

When the imitator observes a new demonstrator action not seen before, the imitator can relate the result of the generating method used to that action. This relation is then placed in the library of correspondences.[3] Using the entries in the library instead of performing the generating method for actions already observed is very often less computationally expensive, especially as the complexity of the algorithm that produces the matching behavior increases. For example, consider a ten degree of freedom robot arm in the real world that has to solve the inverse kinematics equations for moving the manipulator to a point in the workspace, even if that action was performed before from the same initial configuration; solving the inverse kinematics again would be wasteful in such a scenario.

B. Second Component

If only the sequences produced by the generating method were used to build-up the correspondence library, the performance of the imitator would be directly limited by the choice of the algorithm, although possible improvement of the response time required would be observed, as discussed in the previous paragraph. Moreover, some of the stored imitator sequences, although valid solutions to the correspondence problem related to the demonstrator's actions, may become invalid in certain contexts. The second component of ALICE helps to overcome these difficulties: The imitator agent can examine its own history to discover further imitative sequences without having to modify or improve the generating algorithm used. We define this history as a list of actions that were performed so far by the agent while imitating the demonstrator together with these actions' relative effects (also possible effects on the environment). This kind of history provides valuable experience data that ALICE can then use to extract useful mappings to improve and add to the correspondence relation library created up to that point. The methods for actually extracting this information can vary and also managing the sequences that are found can depend on additional metrics (e.g., keep only the shortest sequence that can achieve a particular effect, or keep only the top ten sequences according to some performance measure).

When every possible demonstrator action has been encountered at least once, we can initially say that the library is complete with at least one candidate corresponding imitator sequence for every possible demonstrator action, but such a complete set of correspondence relations between a demonstrator and an imitator cannot necessarily guarantee a consistently satisfying performance of imitation, even in simple environments. A corresponding sequence may be invalid in a different context in which it was observed when it was added to the library. It becomes apparent that as the world resolution and complexity increases, context becomes more relevant and therefore, the variety and quality of the correspondence relations becomes more important. Using the mechanism that extracts sequences from the history as an ongoing feature can address this, as it will continue to enrich the individual mappings with more alternatives that possibly provide better solutions. This second ALICE component relates to Byrne's string parsing theory [14] as it discovers underlying structures that can be used as alternative correspondence solutions on a program-level imitation.

A summary of the ALICE mechanism is given by the following high-level pseudocode:

```
ALICE MECHANISM PSEUDOCODE
Consider the demonstrator behavior as a
  sequence of actions.
For each of these demonstrator actions:
  •  If the demonstrator action has not
  been observed before, create new entry
  in the correspondence library and add
```

[2]Perceptually opaque behaviors [31] are perceived very differently when observed than when being performed, e.g., tongue protrusion or winking, but not singing.

[3]At each stage in its growth, a library of correspondences is an example of a (partial) relational homomorphism between the abstract automata associated to the demonstrator and the imitator [40], [41].

TABLE II
POSSIBLE CORRESPONDENCES FOR DIFFERENT IMITATOR TYPES THAT ARE FOUND USING THE DEFAULT ALGORITHM AS THE GENERATING METHOD (EUCLIDEAN DISTANCE METRIC USED). *Disp*. IS THE RELATIVE DISPLACEMENT EFFECT OF THE ACTION SEQUENCE. GIVEN THE DEMONSTRATOR TYPE, ACTION AND EFFECT, THE POSSIBLE SEQUENCES WITH THEIR EFFECTS FOR EACH IMITATOR TYPE ARE SHOWN ABOVE. NOTE THAT NEITHER THE BISHOP NOR THE KNIGHT CAN ACHIEVE THE EXACT DESIRED DISPLACEMENT

Demonstrator	Action	Disp
Queen	E3	(+3,0)

Imitator	Sequence	Disp.
Queen	[E3]	(+3,0)
Rook	[E3]	(+3,0)
King	[E,E,E]	(+3,0)
Bishop	or [NE,SE] [SE,NE]	(+2,0)
Knight	[N1E2] [S1E2]	(+2,+1) (+2,-1)

sequence of imitator actions found by the generating method.
- If entry already exists, select and use an appropriate action sequence from the correspondence library.

Examine history by considering sequences of actions performed by the imitator so far.

For each of these sequences:
- If the sequence produces same (or perhaps similar) effects to known effects of some known demonstrator action, include sequence as an alternative to the corresponding entry in the library.

In chessworld, the choice of the default algorithm as the imitation sequence generating method was made purely for simplicity and therefore, ALICE, as a generic approach, can augment our initial choice of mechanism instead of replacing it. The generating method is not a very complex one, but as mentioned above, in another setting, the cost of recalculating instead of using an already found solution could be considerable, and the use of a correspondence library is therefore a desirable feature.

In chessworld, the Bishop cannot use the sequence [NE, SE] to imitate the Queen action E3 (see Table II) if the piece is currently located along the northern edge of the board. Instead of using the generating algorithm in such a situation in order to find an alternate solution, it would be desirable already for several possible alternative sequences, e.g., [SE, NE], to be also included in the correspondence library. Note that using any of these two sequences will not move the agent to the desired target square, but instead to an adjacent square. Perfect imitation cannot be achieved for this demonstrator action due to the imitator's embodiment limitations as a Bishop, but if the imitator agent had a Knight embodiment instead, there do exist possible imitating sequences that can achieve such a displacement besides [N1E2] or [S1E2] (see again Table II), e.g., [W1N2, N1E2, N1E2]. This kind of sequence cannot be found using the default-generating algorithm because the value of the metric not only decreases but also increases as a result of certain actions. In Fig. 5, four different possible solutions to the correspondence problem for a Knight imitating a Queen (or Bishop) performing a particular diagonal move are shown. All result in the imitator achieving the same displacement as the imitator, although each follows a different trajectory. Note that the two sequences c) and d) can become invalid if the imitator is too close to the upper or lower edges of the board, respectively, so having many alternative sequences for each entry in the correspondence library can be useful.

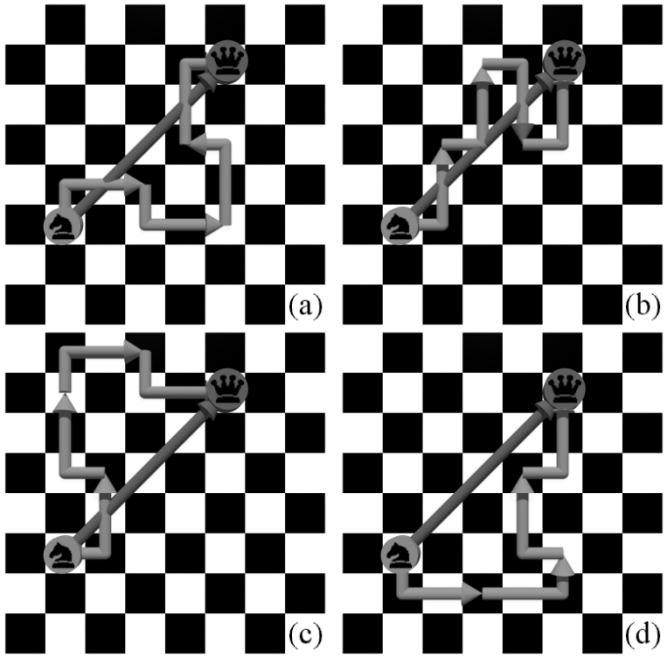


Fig. 5. Four different possible corresponding sequences (a) [N1E2, S1E2, W1N2, E1N2], (b) [E1N2, E1N2, E1S2, E1N2], (c) [E1N2, W1N2, N1E2, S1E2], and (d) [S1E2, N1E2, W1N2, E1N2] that can be used by the Knight to imitate the action NE4 (displacement to the northeast of 4 units) by the demonstrator Queen (or Bishop). These can be found only if ALICE augments the default algorithm.

Fig. 6 shows a possible development of an ALICE correspondence library used by a Knight to imitate a Queen. At each of the time instances shown, every observed demonstrator action is noted as a point at the appropriate vertical and horizontal coordinates of its resulting displacement. These can be either negative or positive relative to the current location of the chess piece. If at least one of the correspondence sequences found so far accomplishes that exact displacement, a dark color tone is used. Otherwise, a light one is used. The shape that slowly emerges relates to the set of actions observed so far and the type of the demonstrator.

V. EXPERIMENTS PERFORMED

This section describes the experiments performed to assess ALICE using the chessworld simulation.

A. Methodology and Experimental Set-Up

In order to test the hypothesis that agents with dissimilar embodiments can improve performance when solving the

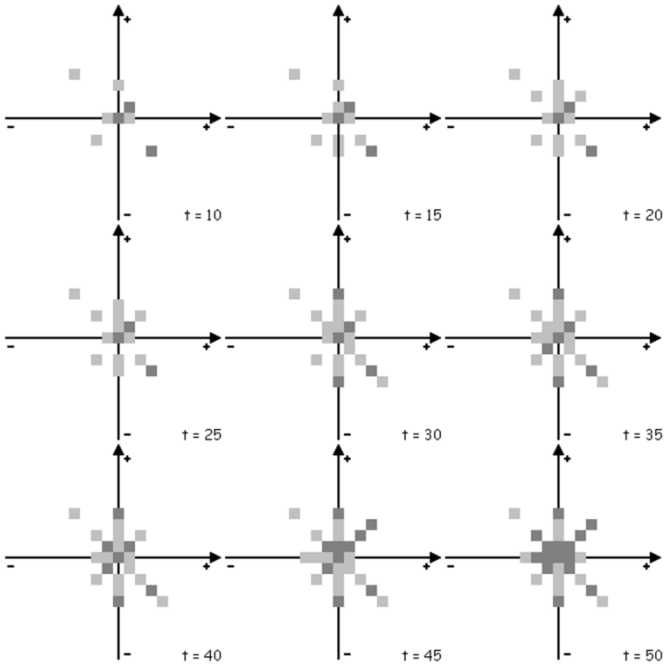


Fig. 6. Possible development of an ALICE correspondence library used by a Knight to imitate a Queen. At every simulation step, the Knight attempts to imitate a Queen sequence of moves. The correspondence build-up is shown in intervals of five simulation steps, left to right, top to bottom. At the displacement coordinates (indicated with respect to horizontal and vertical axes in the figures) or each observed demonstrator action, dark or light color tones indicate whether at least one of the corresponding sequences for this entry satisfies the imitation criteria perfectly or not, respectively. Given that the demonstrator to be imitated is a Queen, the shape that slowly emerges is composed of the vertical, horizontal, and diagonal directions that characterize its movement rules.

correspondence problem in imitation, the chessworld testbed was used to provide dissimilar embodied agents engaging in imitation behavior, with ALICE as the generic mechanism for solving the correspondence problem. Statistical significance in improvement of performance is expected when ALICE is used together with the default algorithm as an improved generating method, compared with using the default generating method on its own.

In each experiment, a demonstrator agent performs a random walk on the chessboard, which other agents must in turn imitate. The demonstrator in every case is embodied as a Queen, which is the chess piece with the least limiting movement rules. This choice was made to give the demonstrator sequences the greatest diverse variety possible among the classic chess pieces.

In the experiments described here, the salient perceptual data are presented to each imitator agent in trajectory-level granularity. This presents a middle ground between simplifying (e.g., end-point-level granularity) and complicating (e.g., path-level granularity) the task. This also takes advantage of the natural segmentation of actions in chess. The length of the demonstrator behavior for each run was 3000 moves. The imitators perceived this random walk segmented into 300 sequences of ten moves.

Both the demonstrator and the imitators start from the same initial square. During the course of each run, because of the dissimilar embodiments, it is possible that displacement errors can appear. The imitator will then have to reach the next target square starting from a different starting point, as the previous target square was not reached exactly. Although these errors affect the performance we chose to allow them for several reasons. First, such unexpected errors in the absolute location of the current square do not change the absolute location of the target square; the problem is only slightly modified. Second, this displacement error can both complicate the imitator's task but also possibly simplify it by placing the agent in a better location (depending on its movement rules).[4] Third, in nature (barring the intervention of a teacher), no resetting or relocation of an imitator occurs during the course of an attempt at imitation.

The metric used in these experiments is the Hamming norm. It was observed that the choice of metric (from among Hamming, Euclidean, and infinity norm) for the algorithm does not affect significantly the quantitative performance measures used by us to evaluate the performance. Qualitatively, however, the character of resulting imitative behaviors does vary with different metrics (see Section III, e.g., Fig. 4).

The following performance measures[5] were used in order to provide an overall view of the imitation attempt.

1) Displacement Performance Measure: The imitation metric used by the default algorithm takes into account distance, so the first performance measure focuses on that aspect of the behavior. It measures the success in minimizing the distance to successive subgoals in a behavioral sequence, using error divided by initial distance to each subgoal, according to the metric, giving an indication of how close to each subgoal the imitator was.

Displacement performance measure is $1-(d_{error}/d_{total})$.
d_{error} is the remaining distance to the current subgoal.
d_{total} is the initial distance to the current subgoal.

2) Subgoal Performance Measure: For each demonstrator sequence of moves, we can measure the number of subgoals achieved over the total number of subgoals. This is an indication of how many subgoals in the behavioral sequence the imitator managed to achieve (depending on the granularity).

Subgoal performance measure is $n_{\text{achieved}}/n_{\text{total}}$.
n_{achieved} is the number of subgoals achieved by the imitator.
n_{total} is the total number of subgoals in this demonstrator sequence.

The implementation of the second component of ALICE requires a specification of a method to extract sequences from the

[4]In order to correct the error and place the agent back on the trail, the imitator would have to use a corrective move that is not part of the normal repertoire. This would add unclassified events in the agent's history and complicates the use of ALICE, since the history would no longer be linear. For ALICE to use such a history, these corrective events would have to be removed first and their effects accounted for, as this corrective type of "move" would not be allowed by the agent's movement rules, and these moves would therefore be invalid for the agent. No such correction method is used here.

[5]Besides the two measures given here, another possible performance measure could be a *speed performance measure*, measuring how many moves the imitator used for each subgoal, against how many moves the demonstrator used. The demonstrator speed for a given sequence depends on the granularity level. For example, in trajectory-level granularity, we have one move per subgoal, in path-level granularity many subgoals per move, and in end-point granularity, many moves per subgoal. In the chessworld scenario, the imitating agents are severely limited by their embodiment and movement rules and since the metrics used by the generating method do not consider at all how fast the chess pieces move, this performance measure is not used here, although it could be used in a slightly modified setting.

imitating agent's history. This is a design issue and can be accomplished in a variety of ways depending on the platform. In the current chessworld scenario, a very simple method is used. At every simulation time step, we consider a moving window of variable length (two to four actions) scanning the history backward. The agent is assumed to have knowledge of the effects for each of its own actions in the particular embodiment repertoire, and the overall effect of each such sequence can therefore be known. If that exact effect is noted as the result of a demonstrator action already present in the correspondence library, then the found sequence is also included in that entry. For example, the Knight is able to discover how to move to adjacent squares in this manner. Note that we require an exact match of effects (displacement in this case) but could also allow for near matches (according to some metric, depending on the action effects considered). Only solutions for previously seen problems will be added, since there is no reason to overload the correspondence library with perhaps unnecessary data—the imitator has no prior knowledge of the complete set of demonstrator actions.

The maintenance of the correspondence library is also deliberately simple in the present scenario. There is no limit to how many correspondences can exist for a single demonstrator action, and no kind of sorting on these corresponding sequences takes place. When the imitator looks up a known demonstrator action, a randomly selected best-so-far solution is returned, if more than one exists.

B. Results and Interpretation of the Results

Each of the plots in the following figures shows the imitation attempts for a random walk of the demonstrator on the chessboard by the different kind of chess pieces. At each simulation step, the agent attempts to imitate a demonstrator move sequence. For each sequence, the average performance measure value of ten simulation runs is plotted. The standard deviation of this measure for these ten runs is shown as a dotted line. A value of 100% shows perfect imitation according to the performance measure used. Plot (a) in every figure shows the performance of the generating method used on its own, and plot (b) shows the performance if the ALICE mechanism is used together with the default-generating method.

1) King and Rook as Imitators: Agents embodied as either King or Rook chess pieces can imitate perfectly the random walks of a demonstrator Queen according to the displacement and subgoal performance measures described above. Both pieces can easily reach every subgoal; the King using repeated single-square moves and the Rook using a pair of horizontal and vertical moves for every diagonal displacement required. The correspondence problem in this case is a relatively simple one and since the imitators can perform perfectly with the default algorithm only, it is not necessary to use the ALICE mechanism.

2) Knight as Imitator: Solving the correspondence problem for the Knight imitating the Queen is more interesting as this embodiment does not prevent the agent from moving to any square on the chessboard, but many of these solutions are impossible to find using only the default algorithm as a generating method.

Figs. 7 and 8 illustrate the improvement in performance (using the displacement performance and subgoal performance measures respectively) that is attained with ALICE. When the agent is using only the default algorithm as the generating method, a partially successful, yet not perfect performance is attained (displacement performance measure average below 80%, subgoal performance measure average below 20%, both not showing improvement). For the same demonstrator behavior, when the ALICE mechanism compliments the generating method and the use of the correspondence library is introduced, eventually, an overall level of generally perfect performance is achieved.

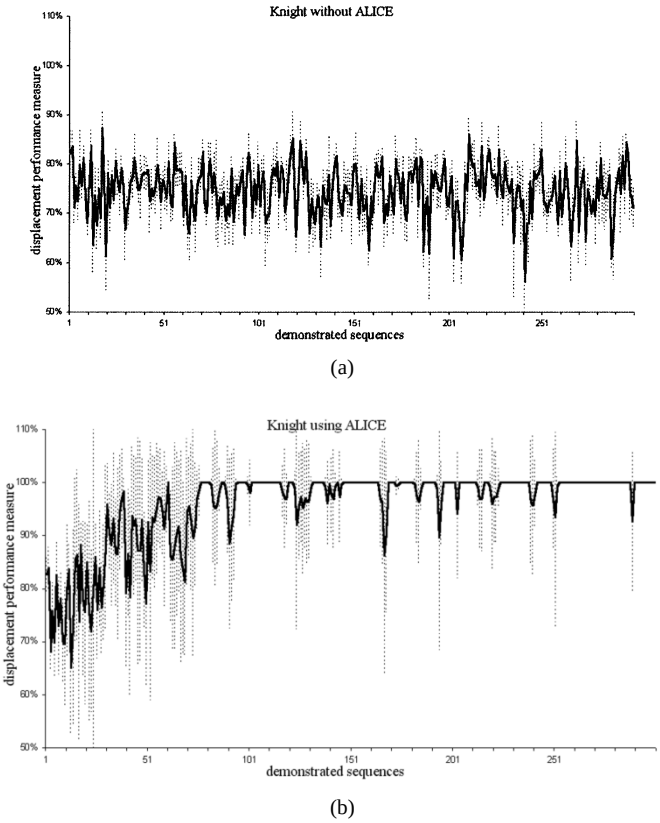


Fig. 7. Displacement performance measure for a Knight imitating a Queen (average for same demonstrator random walks). At each simulation step, the agent attempts to imitate a demonstrator move sequence. For each sequence, the average displacement performance measure value of ten different simulation runs that used the same overall random walk for the demonstrator is plotted. The standard deviation for each sequence is shown as a dotted line. In (b), the ALICE mechanism was used together with the generating method; the default algorithm was used on its own in (a).

Similar experiments were carried out using, instead of ten runs with the same demonstrator random walk, ten runs of different demonstrator random walks for each case. As shown in Fig. 9, the results were similar.

After approximately 75 imitation attempts, the performance difference becomes and remains highly statistically significant (using two-sample t-tests, assuming unequal variances, $p < 0.001$; variances were not homogeneous according to $F - \max$ tests).

Although one would expect a perfect performance level after the correspondence library is complete (in the sense that all possible demonstrator moves have been observed and appropriate correspondence sequences noted), some glitches can be sporadically observed. As noted in Section IV, this is not so unexpected: The chessworld dimensions (eight by eight squares) are restrictive enough to make some of the correspondences invalid

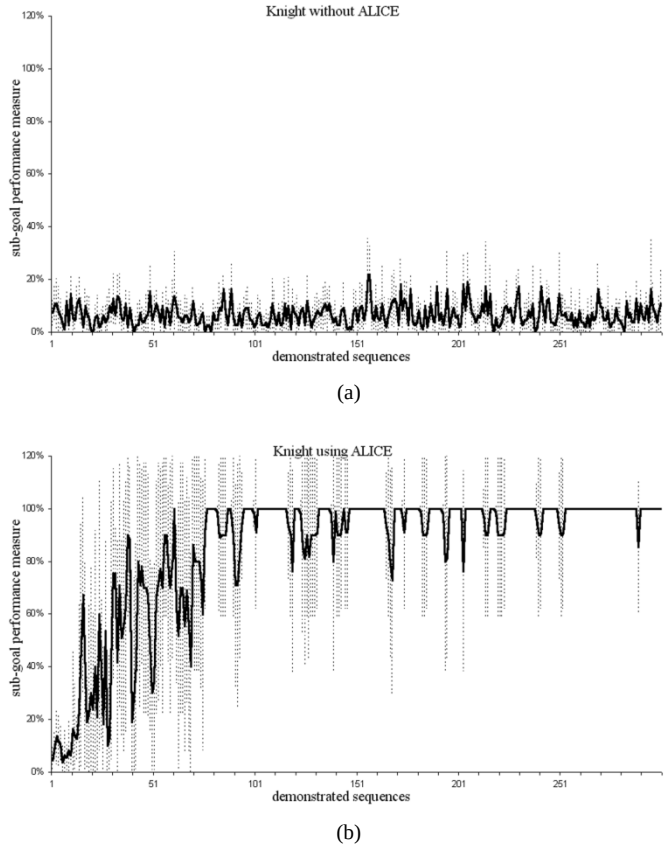


Fig. 8. Subgoal performance measure for a Knight imitating a Queen (average of same demonstrator random walks) (a) without ALICE and (b) using ALICE. For each sequence, the average subgoal performance measure value of ten different simulation runs that used the same overall random walk for the demonstrator is plotted.

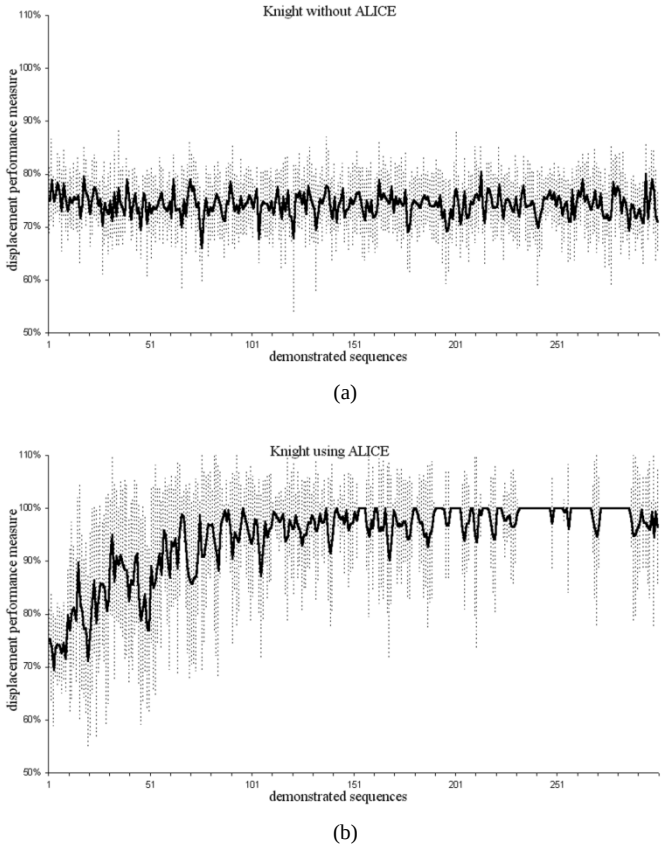


Fig. 9. Displacement performance measure for a Knight imitating a Queen (average for different demonstrator random walks) (a) without ALICE and (b) using ALICE. For each sequence, the average displacement performance measure value of ten different simulation runs that each used a different demonstrator random walk is plotted.

because of the boundaries. These glitches in the performance level seemed to occur when the demonstrator moves close to the edges of the board, and the imitator does not have an appropriate corresponding sequence for these actions in this context. To test that this is the case, we repeated the experiment with the same demonstrator random walk happening in the center of a much larger chessboard so that such situations would not arise, and indeed, the performance level, once the correspondence library has enough perfectly matching entries, remains constantly at 100%. This is shown in Fig. 10.

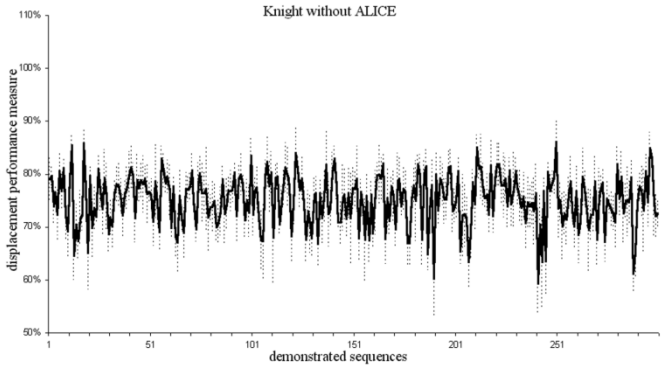
3) Bishop as Imitator: The imitator embodiment as a Bishop chess piece has the major disadvantage that only half the locations on the chessboard can be visited. There are no improved solutions to the correspondence problem that cannot be found by using only the default algorithm as a generating method. The ALICE mechanism yields no noticeable performance improvement (as seen in Figs. 11 and 12) since the demonstrator in the random walk visits a significant amount of inaccessible locations. Performing t-tests showed no statistically significant difference of performance ($p > 0.05$).

VI. Summary of Results

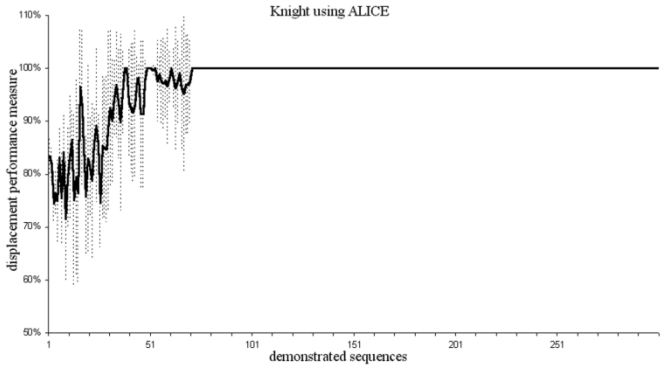
Examining the different chess pieces as imitator agent embodiments established that the ALICE mechanism is useful in solving the correspondence problem when augmenting simple imitation sequence-generating methods. Depending on the specific affordances of the embodiment, the contribution of ALICE in improving the imitating performance can be either significant or only complementary.

The movement affordances of sequences of moves for the imitator King and Rook embodiments allow them to attain any single move displacement achievable by a demonstrator Queen and as such allow a perfect performance according to the performance measures used, although these solutions to the correspondence problem are qualitatively different from the demonstrated model and from each other [see, for example, Fig. 4(b)]. The perfect performance cannot be further improved using the ALICE mechanism, although ALICE can reduce the need to generate the imitating sequences already constructed previously, and instead exploit the correspondence library (see Section IV).

Although the Bishop's embodiment affords a complementary subset of moves to that of the Rook, it is severely disadvantaged in comparison to other pieces/embodiments since it cannot reach half of the positions on the board, whereas all the other pieces can. The generating method can already get the Bishop as close as possible to the target square, and no significant performance difference is noted when ALICE is used, besides the benefits of using a correspondence library, as described earlier.

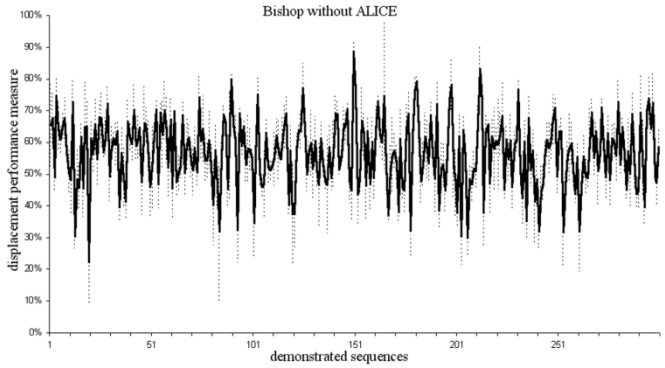


(a)

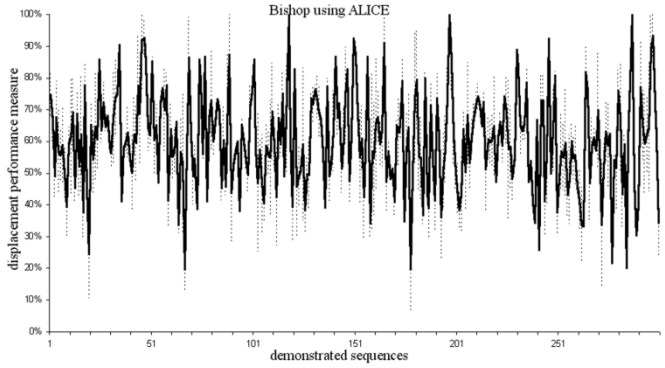


(b)

Fig. 10. Displacement performance measure for a Knight imitating a Queen (unconstrained context, average for same demonstrator random walks) (a) without ALICE and (b) using ALICE. For each sequence, the average displacement performance measure value of ten different simulation runs that used the same overall random walk for the demonstrator is plotted. The same demonstrator random walk as in Figs. 7 and 8 takes place in the center of a 100 $\times$ 100 squares chessboard.



(a)



(b)

Fig. 11. Displacement performance measure for a Bishop imitating a Queen (average for same demonstrator random walks) (a) without ALICE and (b) using ALICE. For each sequence, the average displacement performance measure value of ten different simulation runs that used the same overall random walk for the demonstrator is plotted.

The Knight embodiment is a good example of a seemingly complicated repertoire of moves that nevertheless can successfully mimic all the possible demonstrator actions once the correspondence problem is solved. Many of the demonstrator actions require imitating sequences by the Knight that cannot be found by the default algorithm. ALICE is shown to significantly improve the performance. Once most of the demonstrator actions have been performed and the initial correspondence library entries have been created, we can expect a gradual improvement of performance, resulting in a permanent near-perfect level of performance when all the possible demonstrator actions have been observed and suitable correspondences found, either by the default-generating method or the mechanism that extracts additional correspondence sequences from the history. It is important to have many alternative correspondence sequences for each demonstrator action since some of the imitative sequences may be invalid in a different context from the one in which they were originally observed. The original 8 $\times$ 8 chessboard can be restrictive to that effect and that results in the observed glitches of the otherwise constant performance level of the imitator. If the agents are placed in a larger chessboard where the edges no longer obstruct the performance of the corresponding sequences, the imitator performance level, once a complete correspondence library was attained, was never subject to glitches due to contextual factors.

VII. Conclusions

Imitation and behavioral matching can serve as fundamental components for behavior acquisition both in humans and animals but also in artificial systems. Agent embodiment, together with the use of different metrics and subgoal granularities can affect the success and character of the imitation observed. In order for an agent to successfully imitate, the correspondence problem needs to be solved, finding appropriate mappings between its own actions and the ones of a demonstrator agent with a possibly dissimilar embodiment. It is possible to build up a solution to the correspondence problem incrementally while learning from observing a demonstrator over time. A correspondence serves as a "refractive looking-glass" through which an observed demonstrator's behavior is transformed to yield similar, but possibly not quite the same, action sequences for the imitator. This allows the imitator to get along, using the affordances of its own embodiment, while exploiting observations of the behavior of others in its environment.

We showed how an imitator agent with ALICE exposed to the demonstrator behavior in the chessworld can build up useful partial solutions to the correspondence problem, i.e., mapping the demonstrator actions to those it can perform in its own particular embodiment to achieve similar effects, exhibiting highly successful imitating performance. Effectively, ALICE provides a combination of learning and memory to help solve the corre-

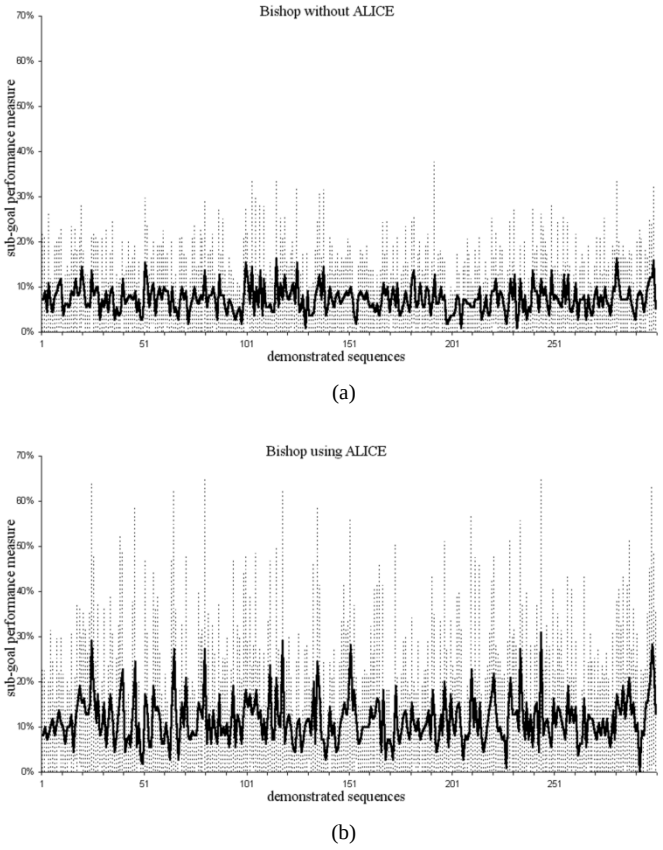


Fig. 12. Subgoal performance measure for a Bishop imitating a Queen (average for different demonstrator random walks) (a) without ALICE and (b) using ALICE. For each sequence, the average subgoal performance measure value of ten different simulation runs that each used a different demonstrator random walk for the demonstrator is plotted.

spondence problem. There is generalization in that learned corresponding action sequences can be reused by the imitator in new situations and contexts. Due to its generic nature, ALICE can be implemented in a variety of ways, not depending on a specific generating method or the platform used. Contrary to the traditional approach in the robotics research on imitation that concentrates on creating dedicated control architectures ignoring the social dimension of imitation, a more general agent-based approach considers the behavior of particular autonomous agents in relation to each other and their environment, using various agent-specific metrics and perceptual saliency criteria. Our work here, while not specifically addressing *interaction*, could form the basis for harnessing imitation in richer social interactions among artifacts or between artifacts and humans. This work could lead to a better understanding of animal imitation or to the improved design of imitation algorithms for robots.

The ALICE mechanism requires a) the addition of a correspondence library and b) that the agent is able to keep track of its actions along with a mechanism to extract additional useful correspondences from its personal history. The contents and development of the correspondence library will depend on the types and effects of actions that the demonstrator and imitator agents are capable of, depending on their embodiments, and also on experience. The methods used in the chessworld implementation to extract useful correspondence sequences from the history were simple, but the methods used in another setting will depend upon the structure of the personal history that the agents keep.

In the chessworld implementation of the ALICE mechanism, both the correspondence library and the imitator agent's history were relatively simple, since there was only a small set of actions (chess moves) with a single type of effect (displacement on the chessboard). The history kept track only of the relative displacements caused by the imitator's own actions. The method to extract correspondences from history involved looking for sequences of bounded length that always perfectly achieve the corresponding effect according to the metric used. The correspondence library was visualized using vertical and horizontal axes on the two-dimensional (2–D) plane, and no sorting or maintenance of the correspondence library entries was needed.

However, in a different environment with more complex agents, a more complex implementation of the ALICE mechanism might be necessary, dealing with actions that may have not only one, but also many diverse type of effects. For example, if the agents not only move in the environment but also interact directly with it (e.g., by being able to pick up objects and move them around), then a more complex implementation of the history is required for keeping track of the imitator actions and all the resulting effects. The mechanism to extract correspondences would have to be able to cope with such a history. The extracted sequences for a correspondence library entry would require some form of sorting when that action correspondence is used, according to a performance evaluation metric.

For future work, we plan to bring ALICE to more complex testbeds, study the requirements for the method that extracts alternative corresponding sequences from the history of the imitator in more complex settings, address issues of perception, segmentation, context, self-repair (via self-imitation of previous optimal behavior), and use more advanced machine learning techniques as components for ALICE.

Scaling up toward more real-world platforms, we describe in [3] some early work in progress implementing the ALICE mechanism on a robotic arm simulation. In this case, the demonstrator and imitator agents are robotic arm manipulators that can have different numbers of joints and/or lengths of these joints, operating in a 2–D workspace.

ACKNOWLEDGMENT

A. Alissandrakis would like to thank Y. Dimitriadi for personal inspiration. The authors would like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Learning how to do things with imitation," in *Proc. Amer. Assoc. Artif. Intell. Fall Symp. Learning How to Do Things*, 2000, pp. 1–6.

[2] ——, "Through the looking-glass with ALICE—Trying to imitate using correspondences," in *Proc. First Int. Workshop Epigenetic Robotics: Modeling Cognitive Development Robotic Syst.*, Lund, Sweden, 2001, pp. 115–122.

[3] ——, "Do as I do: Correspondences across different robotic embodiments," in *Proc. Fifth German Workshop Artif. Life*, Lübeck, Germany, 2002, pp. 143–152.

[4] P. Andry, P. Gaussier, S. Moga, J. P. Banquet, and J. Nadel, "Learning and communication via imitation: An autonomous robot perspective," *IEEE Trans. Syst., Man, Cybern., Pt. A*, vol. 31, pp. 431–442, Sept. 2001.

[5] M. Arbib, "The mirror system, imitation, and the evolution of language," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 229–280.

[6] C. G. Atkeson, J. G. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, E. Kawato, and M. Kawato, "Using humanoid robots to study human behavior," *IEEE Intell. Syst.*, vol. 15, pp. 46–56, Aug. 2000.

[7] A. Billard, "DRAMA, a connectionist model for robot learning: Experiments on grounding communication through imitation in autonomous robots," Ph.D. dissertation, Dept. Artif. Intell., Univ. Edinburgh, Edinburgh, U.K., Oct. 1998.

[8] ——, "Imitation: A means to enhance learning of a synthetic protolanguage in autonomous robots," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 281–310.

[9] A. Billard and K. Dautenhahn, "Experiments in learning by imitation—Grounding and use of communication in robotic agents," *Adapt. Beh.*, vol. 7, pp. 415–438, 1999.

[10] ——, "Grounding communication in autonomous robots: An experimental study," *Robot. Auton. Syst.*, vol. 1–2, no. 24, pp. 71–81, 1998.

[11] A. Billard, G. Hayes, and K. Dautenhahn, "Imitation skills as a means to enhance learning of a synthetic proto-language in an autonomous robot," in *Proc. AISB Symp. Imitation Animals Artifacts*. Edinburgh, U.K., 1999, pp. 88–95.

[12] C. Breazeal and B. Scassellati, "Infant-like social interactions between a robot and a human caretaker," *Adapt. Beh.*, vol. 8, 2000.

[13] R. Brooks, *Cambrian Intelligence*. Cambridge, MA: MIT Press, 1999.

[14] R. W. Byrne, "Imitation without intentionality. Using string parsing to copy the organization of behavior," *Animal Cogn.*, vol. 2, pp. 63–72, 1999.

[15] R. W. Byrne and A. E. Russon, "Learning by imitation: A hierarchical approach," *Beh. Brain Sci.*, vol. 21, pp. 667–709, 1998.

[16] J. Call and M. Carpenter, "Three sources of information in social learning," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 211–228.

[17] A. Cypher, Ed., *Watch What I Do: Programming by Demonstration*. Cambridge, MA: MIT Press, 1993.

[18] K. Dautenhahn, "Trying to imitate—A step toward releasing robots from social isolation," in *Proc. From Perception to Action Conf.*, Lausanne, Switzerland, 1994, pp. 290–301.

[19] K. Dautenhahn, "Getting to know each other—Artificial social intelligence for autonomous robots," *Robot. Auton. Syst.*, vol. 16, pp. 333–256, 1995.

[20] K. Dautenhahn and C. L. Nehaniv, Eds., *Imitation in Animals and Artifacts*. Cambridge, MA: MIT Press, 2002.

[21] K. Dautenhahn and C. L. Nehaniv, "The agent-based perspective on imitation," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 1–40.

[22] J. Demiris and G. Hayes, "Imitative learning mechanisms in robots and humans," in *Proc. Fifth Eur. Workshop Learning Robots*, Bari, Italy, 1996, pp. 9–16.

[23] ——, "Active and passive routes to imitation," in *Proc. AISB Symp. Imitation Animals Artifacts*. Edinburgh, U.K., 1999, pp. 81–87.

[24] E. Furse, "A model of imitation learning of algorithms from worked examples," *Cybern. Syst., Special issue on "Imitation in Natural and Artificial Systems"*, vol. 32, no. 1–2, pp. 121–154, 2001.

[25] V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti, "Action recognition in the premotor cortex," *Brain*, vol. 119, pp. 593–609, 1996.

[26] P. Gaussier, S. Moga, J. P. Banquet, and M. Quoy, "From perception–action loops to imitation processes: A bottom-up approach to learning by imitation," *Appl. Artif. Intell. J., Special issue on "Socially Intelligent Agents"*, vol. 12, no. 7–8, pp. 701–729, 1998.

[27] G. Hayes and J. Demiris, "A robot controller using learning by imitation," in *Proc. SIRS, Second Int. Symp. Intell. Robotic Syst.*, Grenoble, France, 1994, pp. 198–204.

[28] L. M. Herman, "Vocal, social, and self imitation by bottlenosed dolphins," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 63–108.

[29] C. M. Heyes, "Transformational and associative theories of imitation," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 501–524.

[30] C. M. Heyes and B. G. Galef, Jr., Eds., *Learning in Animals: The Roots of Culture*. New York: Academic, 1993.

[31] C. M. Heyes and E. D. Ray, "What is the significance of imitation in animals?'," *Adv. Study Beh.*, vol. 29, pp. 215–245, 2000.

[32] I. L. Kolodner, *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann, 1993.

[33] Y. Kuniyoshi, H. Inoue, and M. Inaba, "Design and implementation of a system that generates assembly programs from visual recognition of human action sequences," in *Proc. IEEE Int. Workshop Intell. Robots Syst.*, 1990, pp. 567–574.

[34] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observations of human performance," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 799–822, Nov. 1994.

[35] H. Lieberman, Ed., "Special issue on 'Programming by example'," in *Commun. ACM*, 2000, vol. 43, pp. 72–114.

[36] M. J. Matarić, "Getting humanoids to move and imitate," *IEEE Intell. Syst.*, vol. 15, pp. 18–24, July/Aug. 2000.

[37] F. Michaud and M. J. Matarić, "Representation of behavioral history for learning in nonstationary conditions," *Robot. Auton. Syst.*, vol. 29, pp. 187–200, 1999.

[38] R. W. Mitchell, "A comparative-developmental approach to understanding imitation," *Perspectives Ethol.*, vol. 7, pp. 183–215, 1987.

[39] J. Nadel and G. Butterworth, Eds., *Imitation in Infancy*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[40] C. L. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Interdisciplinary Approaches to Robot Learning*, Singapore: World Scientific, 2000.

[41] ——, "Like me?—Measures of correspondence and imitation," *Cybern. Syst.: Int. J.*, vol. 32, no. 1–2, pp. 11–51, 2001.

[42] ——, "The correspondence problem," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 41–62.

[43] M. N. Nicolescu and M. J. Matarić, "Learning and interacting in human–robot domains," *IEEE Trans. Syst., Man, Cybern. A*, vol. 31, pp. 419–430, Sept. 2001.

[44] R. Pfeifer and C. Scheier, *Understanding Intelligence*. Cambridge, MA: MIT Press, 1999.

[45] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kauffman, 1993.

[46] G. Rizzolatti and M. A. Arbib, "Language within our grasp," *Trends Neurosci.*, vol. 21, no. 5, pp. 188–194, 1998.

[47] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.

[48] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 171–190.

[49] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends Cogn. Sci.*, vol. 3, pp. 233–242, 1999.

[50] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[51] E. L. Thorndike, "Animal intelligence: An experimental study of the associative process in animals," *Psychol. Rev. Monogr. 2*, pp. 551–553, 1898.

[52] A. Whiten, "Imitation of sequential and hierarchical structure in action: Experimental studies with children and chimpanzee," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA: MIT Press, 2002, pp. 191–210.

[53] T. R. Zentall, "Imitation and other forms of social learning in animals: Evidence, function, and mechanisms," *Cybern. Syst., Special Issue on "Imitation in Natural and Artificial Systems"*, vol. 32, no. 1–2, pp. 53–96, 2001.

Aris Alissandrakis received the M.Eng. degree in cybernetics from the Department of Cybernetics, University of Reading, Reading, U.K., in 1999. He is currently pursuing the Ph.D. degree at the Adaptive Systems Research Group, Computer Science Department, University of Hertfordshire, Hertfordshire, U.K. His research interests include artificial life and imitation in artificial systems.

Chrystopher L. Nehaniv received the Ph.D. degree in mathematics from the University of California, Berkeley, in 1992.

He is currently Professor of mathematical and evolutionary computer sciences at the University of Hertfordshire, Hertfordshire, U.K. He has taught, lectured, and conducted interdisciplinary research at various universities in the United States, Japan, Great Britain, and Hungary. He is Associate Editor of the journal *BioSystems* and Director of the U.K. Engineering and Physical Science Research Council Network on Evolvability in Biological and Software Systems.

Kerstin Dautenhahn received the Ph.D. degree from the Biological Cybernetics Department of the University of Bielefeld, Bielefeld, Germany, in 1993.

She is currently a Reader in artificial intelligence with the Adaptive Systems Research Group at the University of Hertfordshire in England. She previously worked at GMD, St. Augustin, Germany, and the VUB AI-Lab, Brussels, Belgium. Since 1993, she has initiated and led research projects on socially intelligent agents and social robotics. Currently, she directs the Robotics and Interactive Systems Laboratory at the University of Hertfordshire, Hertfordshire, U.K., and is Associate Editor of the journal *Adaptive Behavior*.