Solving the Correspondence Problem Between Dissimilarly Embodied Robotic Arms Using the ALICE Imitation Mechanism

Aris Alissandrakis Chrystopher L. Nehaniv Kerstin Dautenhahn

Adaptive Systems Research Group
Faculty of Engineering & Information Sciences
University of Hertfordshire
Hatfield Herts AL10 9AB
United Kingdom
{A.Alissandrakis, C.L.Nehaniv, K.Dautenhahn}@herts.ac.uk

Abstract

Imitation is a powerful mechanism whereby knowledge may be transferred between agents (both biological and artificial). A crucial problem in imitation is the *correspondence problem*, mapping action sequences of the model and the imitator agent. This problem becomes particularly obvious when the two agents do not share the same embodiment and affordances. This paper describes work with our general imitation mechanism called *ALICE* (Action Learning for Imitation via Correspondence between Embodiments) that specifically addresses the correspondence problem. The mechanism has been implemented in two different software test-beds. The previous implementation, *chessworld*, is briefly summarised and the current *robotic arm manipulator* implementation is presented in this paper.

Using the robotic arm test-bed we present proof of concept for the social transmission of behavioural patterns through groups of heterogeneous agents. We also present experiments that illustrate the impact of *synchronization*, *loose perceptual matching* and *proprioception* on the imitative performance. The robustness and adaptive nature of the ALICE mechanism is further illustrated with examples where the imitator agent embodiment is changing during the initial and later stages of the learning process.

1 Agent-based perspective

Imitation is a powerful learning mechanism and a general *agent-based approach* must be used in order to identify the most interesting and significant problems, rather than the prominent *ad hoc* approaches in imitation robotics research so far. The traditional approach concentrates in 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 (Kuniyoshi et al. 1990, 1994). Model and imitator are usually not interacting with each other, neither do they share and perceive a common context. Effectively this kind of approach limits itself to answering the question of how to imitate for a *particular* robotic system and its *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 to different robot platforms.

In contrast to the above, the agent-based approach for imitation considers the behaviour of an autonomous agent in relation to its environment, including other autonomous agents. The mechanisms underlying imitation are not divorced from the behaviour-in-context, including the social and non-social environments, motivations, relationships among the agents, the agent's individual and learning history etc. (Dautenhahn and Nehaniv, 2002).

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.

In addition to deciding who, when and what to imitate, an agent must employ the appropriate mechanisms to learn and carry out the necessary imitative actions. The embodiment of the agent and its affordances will play a crucial role, as stated in the *correspondence problem* (Nehaniv and Dautenhahn, 2002)*:*

*Given an observed behaviour of the model, which from a given starting state leads the model through a sequence (or hierarchy) of sub-goals 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 sub-goals - in corresponding states, actions, and/or effects, while possibly responding to corresponding events.*

This informal statement[1] of the correspondence problem draws attention to the fact that the agents may not necessarily share the same morphology or may not share access to 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.

In order to study the correspondence problem we developed the *ALICE* (Action Learning via Imitation between Corresponding Embodiments) generic imitation mechanism, and implemented it in different simple test-beds. These test-beds were implemented using the Swarm agent simulation system (`www.swarm.org`).

2 ALICE overview

The imitative performance of an agent with a dissimilar embodiment to the model will not be successful unless the correspondence problem between the model and the imitator actions is (at least partially) solved.

To address this in an easy to generalize way, we developed ALICE (Action Learning for Imitation via Correspondences between Embodiments) as a generic mechanism for building up correspondences based on *any* generating method for attempts at imitation. This mechanism is related to statistical string parsing models of social learning from ethology (Byrne 1999) and also associative sequence learning theory from psychology (Heyes and Ray 2000).

The ALICE mechanism creates a *correspondence library* that relates the actions, states and effects of the model (that the imitator is being exposed to) to actions (or sequences of actions) that the imitator agent is capable of, depending on its embodiment and/or affordances.

These corresponding actions are evaluated according to a metric and can be looked up in the library as a partial solution to the correspondence problem when the imitator is next exposed to the same model action, state or effect. It is very important to note that the choice of metric can have extreme qualitative effects on the imitator's resulting behaviour (Alissandrakis et al, 2002), and on whether it should be characterized as 'imitation', 'emulation', 'goal emulation', etc. (Nehaniv and Dautenhahn, 2002).

The ALICE mechanism is comprised of a generating mechanism, a history mechanism and the correspondence library.

[1] For a formal statement of the correspondence problem relating to the use of different error metrics and for other applications, see also (Nehaniv and Dautenhahn 1998, 2000, 2001).

2.1 The generating mechanism

The model behaviour that the imitating agent is exposed to is segmented as a sequence of actions, states and effects. As the imitating agent is sequentially exposed to these, it has to perform an appropriate action to achieve a behaviour matching that of the model. In this respect, the ALICE mechanism functions as an action controller in an individual agent for achieving imitative behaviour based on its own perceptions and experiences.

The *generating mechanism* is responsible for suggesting candidate actions. Although this component can be complex, in both implementations of ALICE we choose to use a *random* generating mechanism instead. The idea is to accommodate *any* generating mechanism that returns a valid action from the search space. This simple random generating mechanism performs well enough for our test-bed purposes, although the rate of learning is naturally slower than for more complex action generating mechanisms. Sophisticated applications of ALICE can benefit by replacing, in a modular way, this action generating mechanism with a more sophisticated one, appropriate to the given application.

2.2 The history mechanism

Another way of obtaining solutions to the correspondence problem is the *history mechanism*. This mechanism looks for appropriate alternative corresponding sequences of actions for each of the existing correspondence library entries in the history of performed actions so far by the imitator agent. This approach can be useful to overcome possible limitations of the generating mechanism (Alissandrakis et al., 2002).

2.3 Building up the correspondence library

When the imitating agent is exposed to each action, state and effect that comprises the model behaviour, the generating mechanism produces a candidate corresponding action.

If there is no entry in the correspondence library related to the current action, state and effect of the model, a new entry is created, using these as entry keys with the generated action as the (initial) solution[2].

If instead an entry already exists, the new action is compared to the stored action[3]. If the generated action is worse, according to the metric used, then it is discarded and the existing action from the correspondence library is performed. If on the other

[2] More precisely, the contents of the perceptual key depend on the metric the agent is using, for example the keys will only contain states and actions if a composite state-action metric is used.

[3] There is generally more than one stored corresponding action (or sequence of actions) for each entry, reflecting alternative ways to achieve the same result.

hand the new action is better, then it is performed by the agent and the library entry is updated. This could mean that the new action simply replaces the already existing one, or is added as an alternative solution.

Over time as the imitating agent is being exposed to the model agent the correspondence library will reflect a partial solution to he correspondence problem that can be used to achieve a satisfactory imitation performance.

Effectively ALICE provides a combination of learning and memory to help solve the correspondence problem. There is generalization in that the learned corresponding actions (or sequence of actions) can be reused by the imitator in new situations and contexts.

3 The chessworld test-bed

The creation of *chessworld* was inspired by the need to implement a shared environment for interacting agents of different embodiments affording different relationships to the world.

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. 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 chequered board. Note that the actual game of chess is not studied. We simply make use of the familiar context of chess in a generic way to illustrate the correspondence problem in imitation.

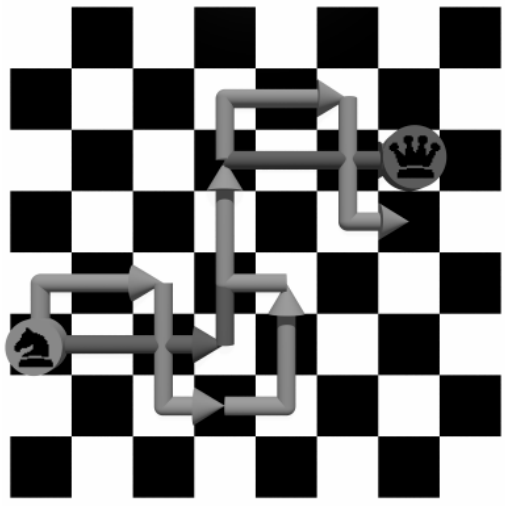

Figure 1: **A chessworld example**. The imitator *Knight* agent is attempting to imitate the movements of the model *Queen* agent.

The range of possible behaviours by the chess agents is limited to movement-related ones. As a model agent performs a random walk on the board, an imitator observes the sequence of moves used and the relevant displacement achieved and then tries to imitate this, starting from the same starting point.

Considering the moves sequentially the agent will try to match them, eventually performing a similar walk on the board. This imitative behaviour is performed after exposure to a complete model behaviour with no obstacles present, neither static (e.g. walls) nor dynamic (e.g. other moving chess pieces), besides the edges of the board which can obstruct movement.

An *action* for a given agent is defined as a move from its repertoire, resulting in a relative displacement on the board. For example a *Knight* agent can perform move `E2N1` (hop two squares east and one square north) resulting in a displacement of `(–2,+1)` relative to its current square.

Addressing *what* to imitate, the model random walk is segmented into relative displacements on the board by using different granularities. For example *end-point level* granularity ignores all the intermediate squares visited and emulates the overall goal (i.e. cumulative displacement) of the model agent. In contrast *path level* granularity not only considers all the squares visited by the model but also the intermediate ones that the chess piece 'slides across' on the chessboard while moving. Between these two extremes, *trajectory level* granularity considers the sequence of relative displacements achieved by the moves of the model during the random walk.

Depending on the embodiment as a particular chess piece, the imitator agent must find a sequence of actions from its repertoire to sequentially achieve each of those displacements.

The assessment of how successful a sequence is in achieving that displacement and moving the agent as close as possible to the target square can be evaluated using different simple geometric metrics (*Hamming norm*, *Euclidean distance* and *infinity norm)* that measure the difference between displacements on the chessboard.

3.1 ALICE in chessworld

For the chessworld implementation ALICE corresponds model actions (moves that result in a relative displacement of the chess piece on the board) to actions (or more probably sequences of actions) that can be performed by the imitator.

The generating mechanism is a random one, returning possible actions from the chess piece moves repertoire of the imitator.

The list of past moves performed by the imitator is defined as the *history*, from which the agent's history mechanism is looking for sequences of actions that can achieve the same relative displacement as model action entries in the correspondence library. The history mechanism is used in parallel to take advantage of this experiential data, compensating for the generating mechanism not allowing moves that locally might increase the distance, but globally reduce the error, within the generated sequences.

The success and character of the imitation observed can be greatly affected by agent embodiment, together

with the use of different metrics and sub-goal granularities.

For a more detailed description of chessworld and the ALICE implementation in this test-bed, see (Alissandrakis et al, 2002).

4 The Robotic Arm test-bed

The robotic arm test-bed was created as a simple, yet "rich enough" environment that would allow for several interacting model and imitator agents, having dissimilar embodiments to each other. Each agent (see Fig. 2) occupies a two-dimensional workspace and is embodied as a robotic arm that can have any number of rotary joints, each of varying length. The agent embodiment can be described as the vector $L = [l_1\ l_2\ l_3\ \ldots\ l_n]$, where l_i is the length of the i^{th} joint.

There are no complex physics in the workspace and the movement of the arms is simulated using simple forward kinematics but without collision detection or any static restraints (in other words, the arms can bend into each other). Our intention is to demonstrate the features of the imitative mechanism and not to build a faithful simulator.

An *action* of a given agent is defined as a vector describing the change of angle for each of the joints, $A = [\alpha_1\ \alpha_2\ \alpha_3\ \ldots\ \alpha_n]$, where *n* is the number of joints. These angles are relative to the previous state of the arm and can only have three possible values, +10° (anti-clockwise), 0° or -10° (clockwise).

A *state* of an agent is defined as the absolute angle for each of the joints, $S = [\sigma_1\ \sigma_2\ \sigma_3\ \ldots\ \sigma_n]$, where *n* is the number of joints. We can distinguish between the *previous state* and the *current state* (the state of the arm after the current action was executed). As a result of the possible actions, the absolute angle at each joint can be anywhere in the range of 0° to 360° (modulo 360°), but only in multiples of 10°.

The end tip of the arm can leave a trail of paint as it moves along the workspace. The *effect* is defined as a directed straight line segment connecting the end tip of the previous and the current states of the arm (approximating the paint trail). The effect is implemented as a vector of displacement $E = (x_c - x_p, y_c - y_p)$, where (x_p, y_p) and (x_c, y_c) are the end tip coordinates for the previous and current state respectively.

The model pattern is broken down as a sequence of actions that move the robotic arm of the agent from the previous state to the current state, while leaving a behind a trail of paint as the effect.

The nature of the experimental test-bed with the fixed base rotary robotic arms favours circular looping effects and the model patterns used in the experiments were designed as such (see Fig. 3).

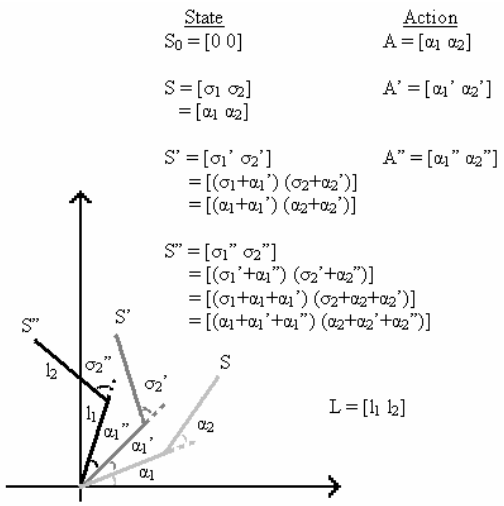

Figure 2: **Example embodiment**. A two-joint robotic arm with arms of length l_1 and l_2, moving from state S_0 (arm completely outstretched along the horizontal axis) to state *S* to state *S'* to state *S"*, as it sequentially performs actions *A*, *A'*, and *A"*. Note that the effects are not shown in this figure.

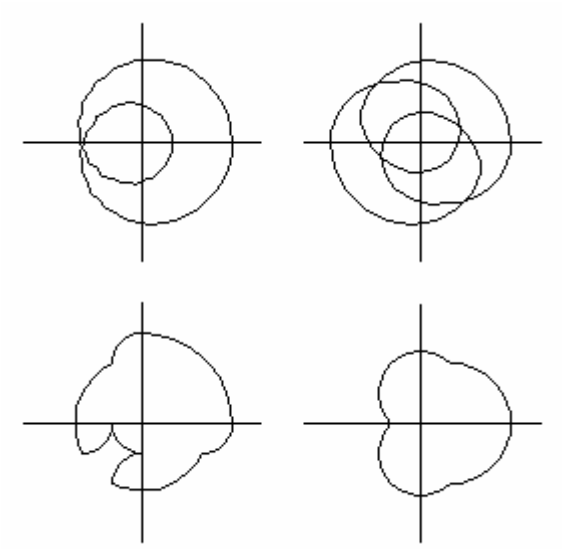

Figure 3: **Four different examples of model behaviours**. Shown are the effect trails created by the end tip of the model agent manipulator arm after a complete behavioural pattern. All model agents shown have the same embodiment `L=[15 15 15]`.

Each complete behavioural pattern that returns the arm to its initial state observed by the imitator is called an *exposure*, and the imitator is exposed to repeated instances of the same behavioural pattern. At the beginning of each new exposure it is possible to reset the imitating agent to the initial state. This resetting is called *synchronization* in our experiments.

4.1 Metrics

The imitating agents can perceive the actions, states and effects of the model agents, and also their own actions, states and effects, and therefore we define several metrics to evaluate the similarity between them. Ideally the metric value should be zero, indicating a perfect match.

4.1.1 State metric

The state metric calculates the average distance between the various joints of an agent (posed in a particular state) and the corresponding joints of another agent[4] (posed in a different state) as if they were occupying the same workspace. Ideally this distance should be zero when the arms take corresponding poses, but this may not be possible due to embodiment differences. Using forward kinematics, the coordinates of the ends for each joint are found.

$$x_i = \sum_{j=1}^{i-1} x_j + l_i \cos\left(\sum_{j=1}^{i} \sigma_j\right)$$
$$y_i = \sum_{j=1}^{i-1} y_j + l_i \sin\left(\sum_{j=1}^{i} \sigma_j\right)$$

If both agents have the same number of joints the correspondence between them is straightforward; the Euclidean distance for each pair is calculated, the distances are then all summed and divided by the number of joints to give the metric value.

$$d_i = \sqrt{\left(x_i^{Model} - x_i^{imitator}\right)^2 + \left(y_i^{Model} - y_i^{imitator}\right)^2}$$
$$\mu = \frac{\sum_{i=1}^{n} d_i}{n}$$

If the agents have a different number of joints, then some of the joints of the agent with more are ignored. To find which joint corresponds with which, the *ratio* of the larger over the smaller number of joints is calculated, and if not integer, is rounded to the nearest one. The i^{th} joint of the agent with the smaller number of joints, will correspond to the (ratio × i)th joint of the agent with the larger number of joints. For example if one of the agents has twice the number of joints, only every second joint will be considered.

4.1.2 Action metric

For the action metric, the same algorithm as the one described above for the state metric is used, but considering the action vectors instead of the state vectors.

The value in the case of the state metric represents an absolute position error; for the action metric, it represents the relative error between the changes of the state angles, due to the compared actions.

4.1.3 Effect metric

The effect metric is defined as the Euclidean length $\mu = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ of the vector difference between two effects (x_1, y_1) and (x_2, y_2).

4.2 ALICE implementation

The robotic-arm test-bed is more complex than chessworld, and as a result the implementation of ALICE was adapted to reflect this.

At each time step, the imitator agent may perceive the action, previous and current state and also the effect of the model agent. The imitator might perceive any of those aspects or a combination.

When created, each entry in the correspondence library can contain the action/state/effect of the observed model agent and the current state of the imitator, as perceptual and proprioceptive components respectively, as the key for that entry.

A random action (a different one is generated every time) can be initially used as the attempted corresponding action. It is possible to replace this part of ALICE with a more complex generating mechanism (i.e. inverse kinematics), but the idea is to have a mechanism that simply returns valid actions from the search space. In order to speed up the learning, it is possible to generate more than one random action and choose a best one.

If the model's action triggers a perceptual key, e.g. if it has been observed before, then there is also at least one corresponding action in that library entry. Controlled by a threshold, it is possible not to require an *exact* match for the perceptual and/or the proprioceptive components of the trigger key, but a loose one that is "close enough". We call this *loose perceptual matching* and we hypothesized that it should support learning and generalization.

The two actions (the newly generated one and the best one found in the correspondence library) are then evaluated and compared to each other according to a metric. Depending on which action scored better, the imitator agent will perform that action and the correspondence library entry will be updated accordingly. If there is no matching entry in the library, a new one is created and the new random action is performed. In the current implementation, each entry can store up to three possible corresponding actions that can be seen as possible alternatives.[5]

[4] The state metric can be used not only between different agents, but also to evaluate the similarity between two states of the same agent. This is true for the action and the effect metric as well.

[5] Note that the history mechanism which also considers sequences of past imitative attempts when updating the correspondence library entries is not implemented in the robotic arm test-bed since simple action to action correspondence suffices here. In contrast, corresponding

5 Experiments on aspects of imitation

Using the robotic arm test bed we conducted various experiments to study the possibility of social transmission of behaviours through heterogeneous agents, the affect of proprioception, loose perceptual matching and synchronization on the imitation learning performance, and also the robustness of the ALICE mechanism when the imitator embodiment changes during the learning process, and also after achieving a successful imitative performance.

5.1 Cultural transmission of behaviours

Besides being a powerful learning mechanism, imitation broadly construed is required for cultural transmission (e.g., Dawkins 1976). Transmission of behavioural skills by social learning mechanisms like imitation may also be fundamental in non-human cultures, e.g. in chimpanzees (Whiten et al. 1999), whales and dolphins (Rendell and Whitehead, 2001).

The robotic arm test-bed makes it possible to study examples of behavioural transmission via imitation, with an imitator agent acting as a model for another imitator. If the original model and the final imitator have the same embodiment but the intermediate imitator a different one, we can look at how the different embodiment and the choice of metrics for the evaluation of a successful imitation attempt can affect the quality of the transmitted behaviour.

The example shown in Fig. 4 shows such a transmission of the original model behaviour via an intermediate agent. Although the intermediary has a different embodiment, the original model and final imitator have the same embodiment, and the model behavioural pattern was transmitted perfectly. This was partially helped by the use of the action metric for evaluation to overcome the dissimilar embodiment of the transmitting agent. This example serves as proof of the concept that by using social learning and imitation, rudimentary cultural transmission with variability is possible among robots, even heterogeneous ones.

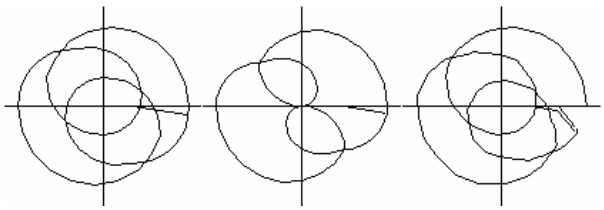

Figure 4: **An example of social transmission**. The original model (`L=[20 20 20]`) is shown to the left. In the middle, an imitator (`L=[30 30]`) acts also as a model for the imitator on the right (`L=[20 20 20]`). Both imitators use the action metric.

sequences of actions are necessary in chessworld as most chess pieces are unable to move as far as their model using only a single action.

The choice of metrics and the particular embodiment of the agents greatly affect the qualitative aspects of imitation, making not every combination suitable for passing on model behaviours, besides crucial aspects of the model behaviours themselves. Note that in Fig. 4, the intermediate agent imitates qualitatively differently, due to its dissimilar embodiment. If the particular embodiment of the intermediate agent greatly distorts the model pattern, then such a transmission might be impossible.

5.2 Synchronization

At the end of each exposure of the imitating agent to the model, it is possible to reset the imitator arm to the same initial position, as a result synchronizing the imitation attempt to the model behaviour.

We conducted ten experimental runs, each with two imitating agents trying to imitate a model agent, one of them synchronizing with the model by resetting to the initial outstretched initial state after the completion of each exposure, and the other starting each attempt from the final reached state of the previous attempt (ideally the same as the initial state, as all the model patterns are designed as closed loops). Both model and imitator agents had the same embodiment (`L=[20 20 20]`) and the metric used was a weighted half-half combination of the action and state metrics. Both imitating agents use proprioception and allow for a 10% margin of looseness for matching the trigger keys. The generating mechanism was creating five random actions to choose from. Each run lasted twenty exposures and the maximum metric value for each exposure was logged.

The ratio of the maximum error of the imitating agent that uses synchronization over the maximum error of the agent that does not reset back the start position at the end of each exposure can be seen in the bottom panel of Fig. 5, constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator, indicating that it is very difficult for an imitating agent that does not synchronize to reach again states relevant to the model pattern if the initial imitation attempts are not successful. This reduces the chance to update and improve the relevant correspondence library entries as the agent wanders with no point of reference. If the state space is large enough, it is possible for the agent to get completely lost.

5.3 Proprioceptive matching

The correspondence library entry keys can contain both perceptive (the action, state and effect of the model agent) and proprioceptive (the imitator's own state at the time of the observation) data. It is possible to ignore the prioperception and trigger the keys based only on the perception.

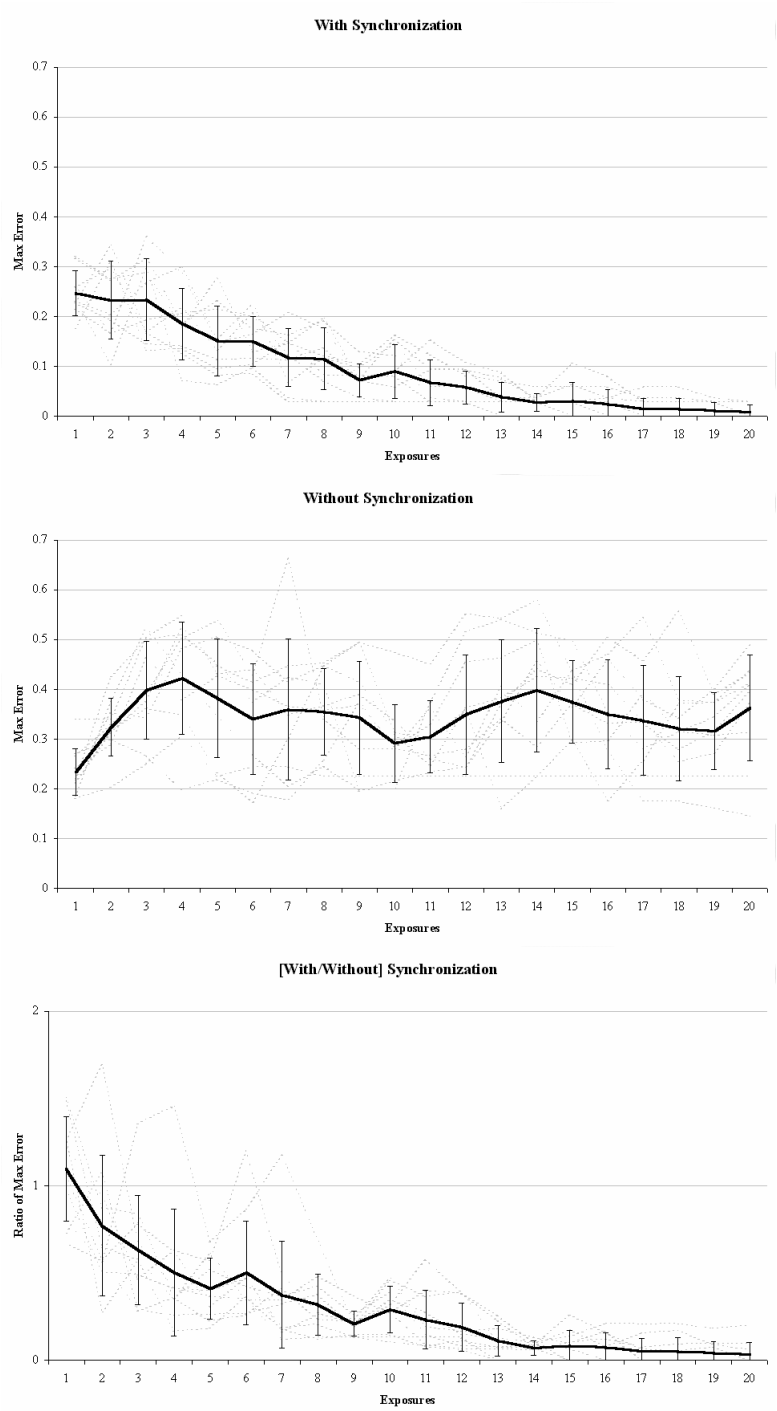

Figure 5: **Experiments comparing the use of synchronization**. The average maximum error metric value of robotic agents over 20 exposures using synchronization (top panel) vs. not using synchronization (middle panel). The ratio of the maximum error per exposure of the imitating agent using synchronization over the maximum error of the imitating agent that does not use synchronization (bottom panel) indicates a comparative many-fold reduction of error with use of synchronization. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment `L=[20 20 20]` and the imitator agents use a half-half composite of the action and state metrics. Both imitators use proprioception and allow for 10% loose perceptual matching.

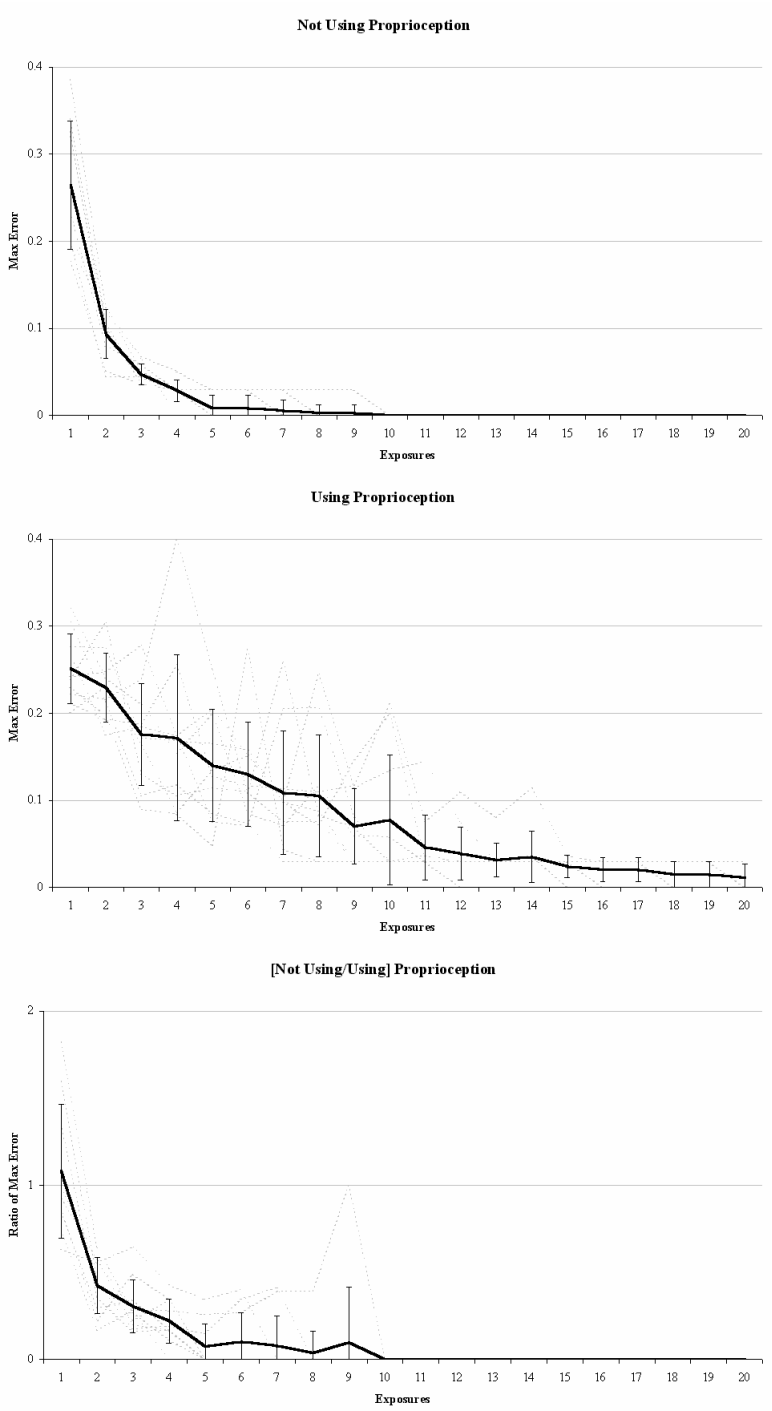

Figure 6: **Experiments comparing using and not using proprioception.** The maximum error metric value of robotic agents over 20 exposures not using proprioception (top panel) vs. using proprioception (middle panel) when searching through the correspondence library entry keys. The ratio of the maximum error per exposure of the imitating agent not employing proprioception over the maximum error of the imitating agent that does (bottom panel) indicates some comparative reduction of error when not using proprioception. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment `L=[20 20 20]` and the imitator agents use a half-half composite of the action and state metrics. Both imitators synchronize and allow for 10% loose perceptual matching.

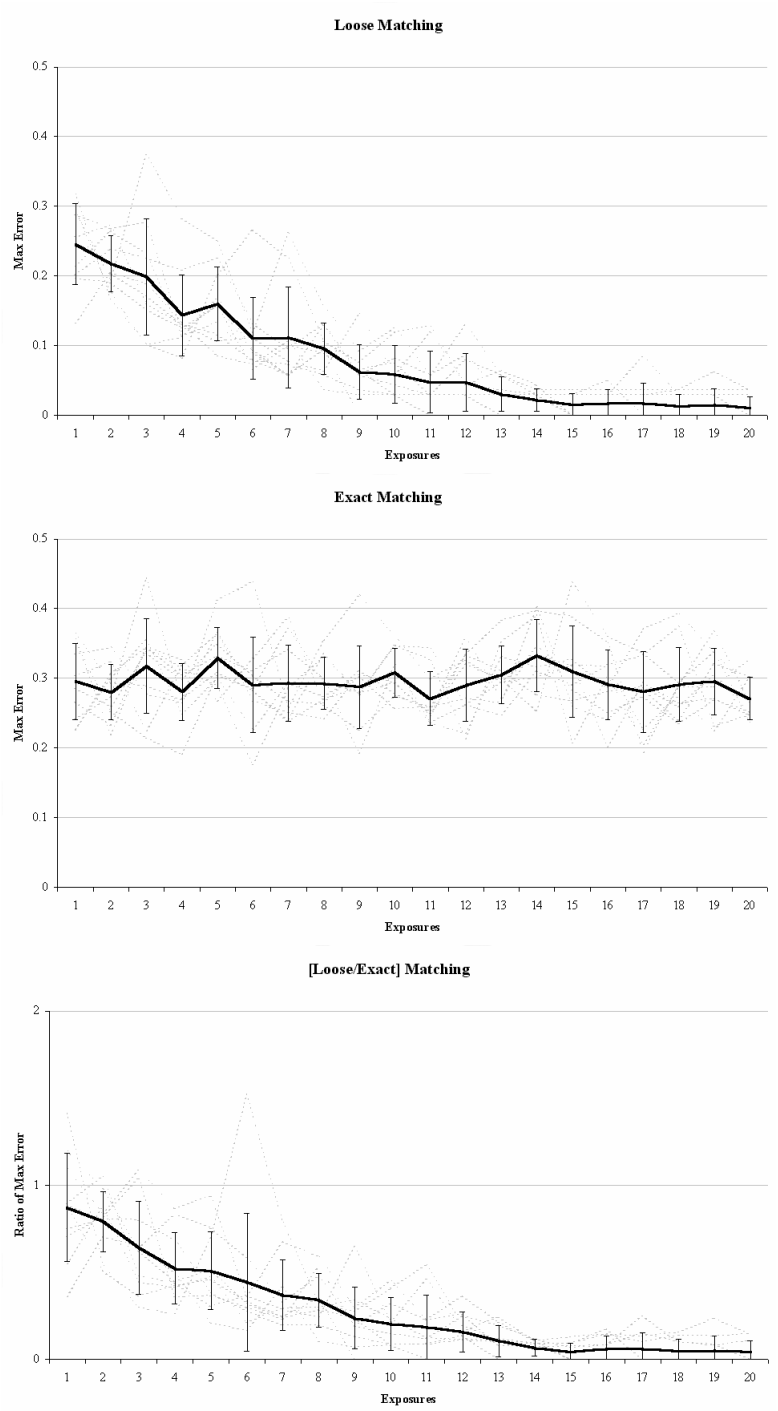

Figure 7: **Experiments comparing the use of loose perceptual matching**. The average maximum error metric value of robotic agents over 20 exposures using loose matching (top panel) vs. using exact matching (middle panel). The ratio of the maximum error per exposure of the imitating agent using loose matching over the maximum error of the imitating agent that uses exact matching (bottom panel) indicates a comparative many-fold reduction of error with use of loose matching. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment `L=[20 20 20]` and the imitator agents use a half-half composite of the action and state metrics. Both imitators synchronize and use proprioception.

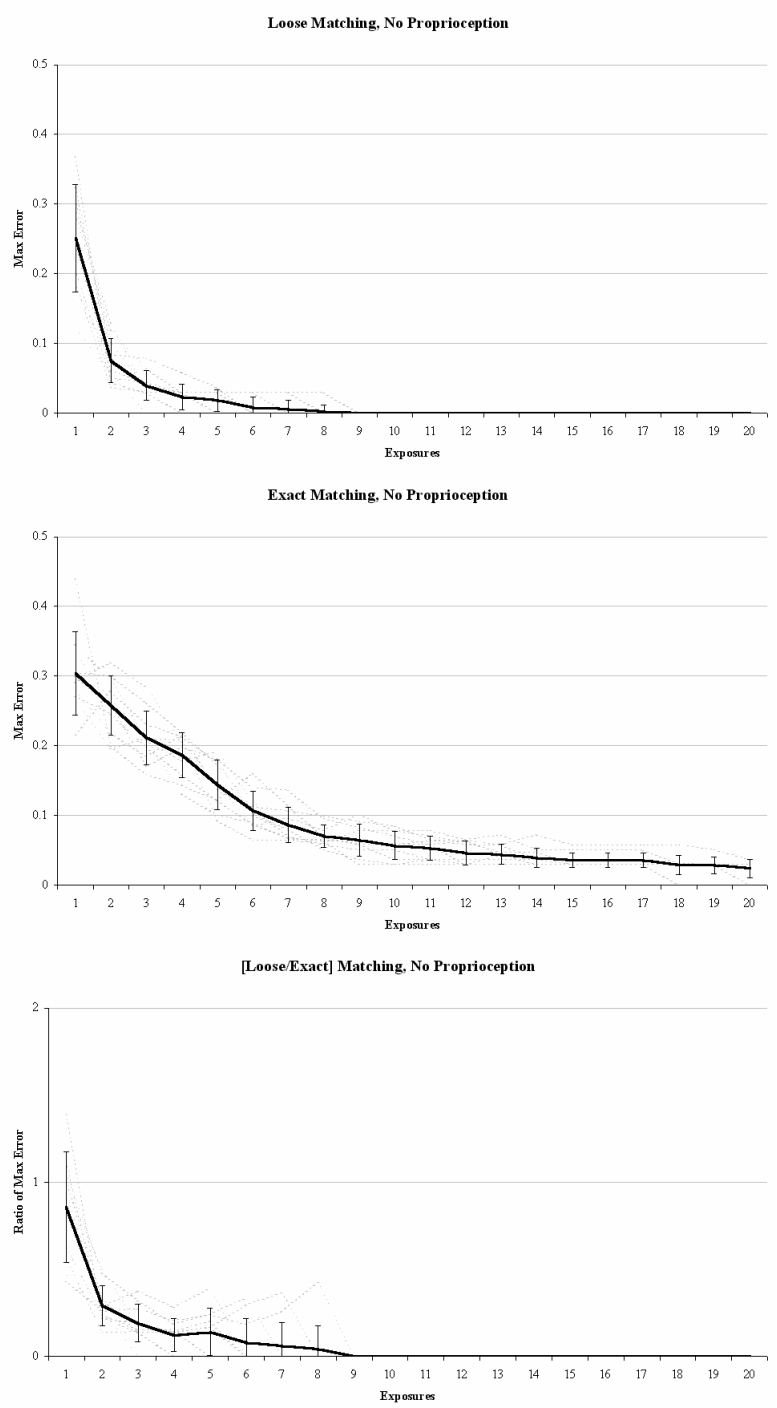

Figure 8: **Experiments comparing the use of loose matching without proprioception**. The average maximum error metric value of robotic agents over 20 exposures using loose matching (top panel) vs. using exact matching (middle panel), both without using proprioception. The ratio of the maximum error per exposure of the imitating agent using loose matching over the maximum error of the imitating agent that uses exact matching (bottom panel) indicates some comparative reduction of error with use of loose matching. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment `L=[20 20 20]` and the imitator agents use a half-half composite of the action and state metrics. Both imitators synchronize and do not use proprioception.

We conducted ten experimental runs, each with two imitating agents trying to imitate a model agent, one of them using proprioception, the other not. Both model and imitator agents had the same embodiment (`L=[20 20 20]`) and the metric used was a weighted half-half combination of the action and state metrics. Both imitating agents used a loose perceptual matching of 10% (see section 5.4 below) and the generating mechanism was creating five random actions to choose from. Each run lasted twenty exposures and the maximum error metric value for each exposure was logged.

The ratio of the maximum error per exposure of the imitating agent that does not use proprioceptive matching over the maximum error of the imitating agent that does can be seen in Fig. 6 (bottom panel), constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator. This indicates that ignoring the proprioceptive component improves the performance rate.

Ignoring the proprioceptive component of the entry keys will confine the number of entries only to the number of different actions, states and effects that define each model pattern, resulting in a much smaller search space. This reduced number of entries in the correspondence library will have the opportunity to update and improve more often, and explains the performance rate improvement. However given enough time, it is expected that proprioception would allow the imitator to eventually learn much finer control in distinguishing appropriate choices of matching actions depending on its own body state.[6]

5.4 Loose perceptual matching

When the ALICE mechanism looks in the correspondence library to find the relevant entry to the currently perceived model actions, states and effects, it is possible not to require an exact match of the entry keys, but one that is close enough, depending on a threshold.

We conducted ten experimental runs under the same conditions. Each run consisted of twenty exposures to the model behaviour for two imitating agents, one of them accepting a 10% margin of looseness for the trigger keys and the other one requiring an exact match, both using proprioception. Model and imitator agents have the same embodiment (`L=[20 20 20]`) and the metric used was a weighted half-half combination of the action and state metrics. The generating mechanism for the imitating agents was creating five random actions to choose from. The maximum metric value for each exposure was logged and is shown in Fig. 7, using loose matching (top panel) and exact matching (middle panel).

The ratio of the maximum error of the agent that uses loose over the agent that uses exact matching can be seen in the bottom panel of Fig. 7, constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator, showing a faster improvement of performance for the imitator agent using loose matching.

Examining the middle panel of Fig. 7, there is no obvious performance improvement in this early stage of learning, although the same amount of time is enough to minimize the error for the agent using a loose matching in the top panel. This is mostly due to the large number of entries created in the correspondence library due to the different proprioceptive states that the agent visits during the imitation attempts. The exact match requirement will create a large number with the same perceptive but different proprioceptive part of the keys.

To illustrate the influence of loose matching on the imitation performance separately from the influence of proprioception, we conducted ten additional experimental runs with the same conditions as the ones described above, but with both imitator agents not using proprioception. The middle panel of Fig. 8 (showing the maximum error for agents requiring exact matching for the perception but ignoring the proprioception part of the trigger keys) indicates that there is a now faster improvement of performance, but still slower compared to the top panel (showing the maximum error for imitating agents allowing a 10% margin of looseness, and not using proprioception). The bottom panel showing the ratio of the maximum errors confirms that loose matching improves the rate of the imitative performance.

5.5 Changes in the agent embodiment

For each agent, vector *L* defines its embodiment, the number of arm segments and their lengths. We define a *growth vector G*, of same size as *L*. By adding (or subtracting) these two vectors we get *L'*, a new embodiment with modified joint lengths, simulating the development of the agent. The growth vector can either increase or reduce the length for each of the joints. The number of joints must remain constant because such a change makes any existing contents of a correspondence library invalid[7].

The growth vector can be used to simulate the body development of the imitator agent during the learning process. One such example is shown in Fig. 9. Although the imitator constantly changes embodiment, starting from half the size and finally reaching the same size as the model agent, the learning process is not affected, resulting in a successful imitation performance.

[6] In this implementation, using proprioception increases the size of the search space by a factor of 36 to the n^{th} power, where n is the number of joints in the imitator.

[7] A robotic arm with a different number of joints would not be able to perform the stored actions, as they describe the angle changes for each of the existing arm joints when those actions were created.

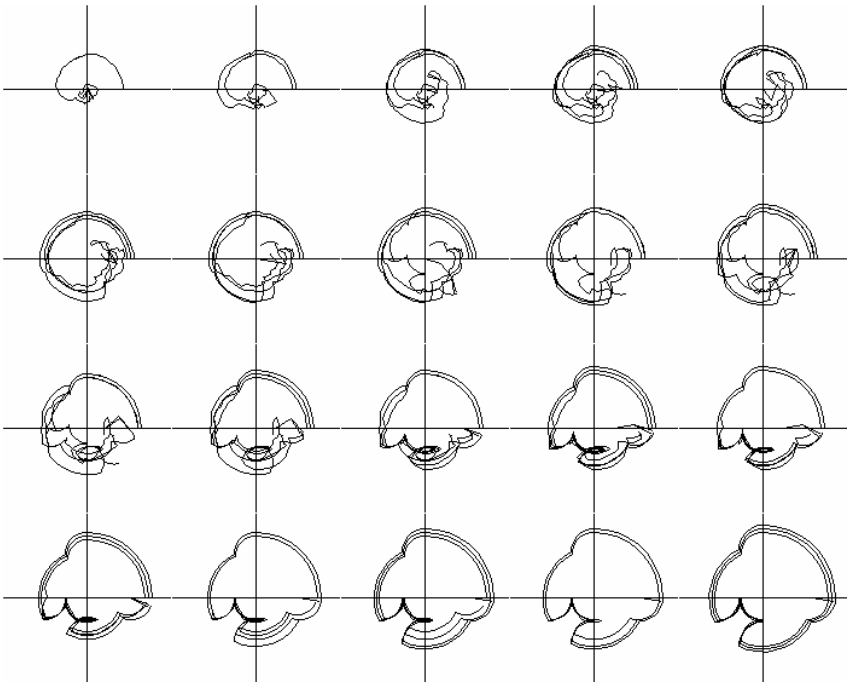

Figure 9: **An example of a growing agent successfully learning to imitate a model pattern.** The figure shows twenty consecutive exposures (left to right, top to bottom). The imitator agent starts on the top left with an initial embodiment `L=[10 10 10]`, and uses a growth vector `G=[1 1 1]` after each exposure to grow up to the embodiment `L=[20 20 20]` of the (unchanging) model agent. The action metric is used, synchronization, proprioception and 10% loose perceptual matching. The effects of the previous 4 attempts are also shown.

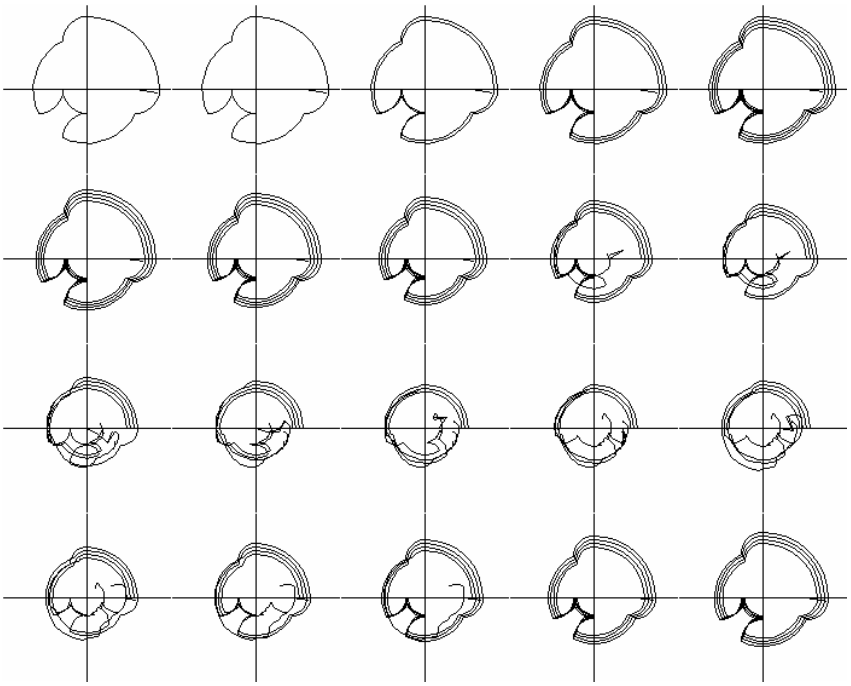

Figure 10: **An example of embodiment changes after successful learning.** The figure shows 20 consecutive exposures (left to right, top to bottom) of an imitator agent that starts on the top left already capable to successfully imitate the model pattern. Starting from an embodiment `L=[20 20 20]`, a growth vector `G=[1 1 1]` is used after each exposure to initially reduce and then expand the embodiment back to the original size. The imitator uses the action metric, synchronization, proprioception and 10% loose perceptual matching. The effects of the previous 4 attempts are also shown.

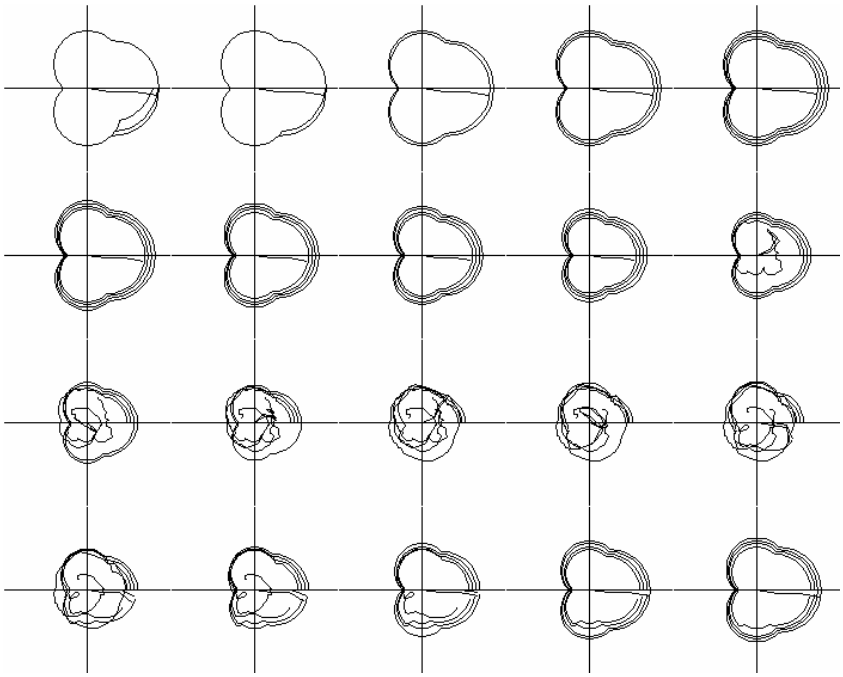

Figure 11: **Another example of embodiment changes after successful learning.** The figure shows 20 consecutive exposures (left to right, top to bottom). The imitator agent that starts on the top left already capable to successfully imitate the model pattern. Starting from an embodiment `L=[20 20 20]`, a growth vector `G=[1 1 1]` is used after each exposure to initially reduce and then expand the embodiment back to the original size. The action metric is used, synchronization, proprioception and 10% loose perceptual matching. The effects of the previous 4 attempts are also shown.

Two examples of using a growth vector to alter the embodiment of the imitator agent are shown in Figs. 10 and 11. In both examples the imitator starts already been capable to imitate a model pattern. During an initial learning stage (not shown in the figures), both imitator agents had the same constant embodiment as their respective model agents (`L=[20 20 20]`). While still being exposed to the model, the lengths of the joints are first reduced and then increased back to their original size. Although the embodiment changes, the agent is able to continually update the contents of the correspondence library to compensate. The imitation performance breaks down when the joint lengths are reduced beyond a certain point, but the ALICE mechanism is robust enough to allow recovery when the agent starts to grow again.

The metric used in both cases is the action metric, compensating for the large range of dissimilar embodiments, and the difference in what they afford. As mentioned in section 5.1 above, the choice of metrics greatly affects the character and quality of the imitation, especially between dissimilar embodiments. For example if the effect metric is used instead of the action metric, very poor results are observed, as the paint strokes created by the shorter joints cannot successfully compensate for the longer strokes achieved by the longer arms of the reference model. In contrast, a robotic arm can equally well rotate clockwise, independent of its length.

These examples show that the ALICE mechanism can be robust enough (with a certain tolerance) to compensate for embodiment changes during the learning stage and after.

6 Conclusions

The results of our experiments using ALICE in the two test-beds described, and particularly in the robotic arm test bed that is presented in greater detail in this paper show that:

1. Cultural transmission is possible in a heterogeneous community of robots via imitation,
2. Loose perceptual matching increases the rate of solving the correspondence problem significantly,
3. Synchronization dramatically increases the rate of solving the correspondence problem,
4. Proprioceptive matching does not seem, at least in the early stages of learning, to aid in the solution of this problem in terms of learning rate.
5. The ALICE imitation mechanism is shown in examples to be reasonably robust to adapt to embodiment changes a) during the early learning process and also b) after the imitating agent has successfully learned how to imitate the model behaviour.

References

Aris Alissandrakis, Chrystopher L. Nehaniv and Kerstin Dautenhahn, Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Trans. Systems, Man & Cybernetics: Part A*, 32(4):482-496, 2002.

Richard W. Byrne. Imitation without intentionality: using string parsing to copy the organization of behaviour. *Animal Cognition*, 2:63-72, 1999.

Kerstin Dautenhahn and Chrystopher L. Nehaniv. The agent-based perspective on imitation. In Kerstin Dautenhahn and Chrystopher L. Nehaniv, editors, *Imitation in Animals and Artifacts*. MIT Press, 2002

Richard Dawkins. *The Selfish Gene*. Oxford, 1976.

Cecilia M. Heyes and Elizabeth D. Ray. What is the significance of imitation in animals? *Advances in the Study of Behavior*, 29:215-245, 2000.

Yasuo Kuniyoshi, Hirochika Inoue and Masayuki Inaba. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. *Proc. IEEE Int. Workshop Intell. Robots Syst.* pages 567-574, 1990

Yasuo Kuniyoshi, Masayuki Inaba and Hirochika Inoue. Learning by watching: Extracting reusable task knowledge from visual observations of human performance. *IEEE Trans. Robot. Automat.* 10:799-822, 1994

Chrystopher L. Nehaniv and Kerstin Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In John Demiris and Andreas Brik, editors, *Proceedings European Workshop on Learning Robots 1998 (EWLR-7)*, Edinburgh, 20 July 1998, pages 64-72, 1998.

Chrystopher L. Nehaniv and Kerstin Dautenhahn. Of Hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In John Demiris and Andreas Brik, editors, *Interdisciplinary Approaches to Robot Learning*, pages 136-161. World Scientific Series in Robotics and Intelligent Systems, 2000.

Chrystopher L. Nehaniv and Kerstin Dautenhahn. Like me? – Measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11-51, 2001.

Chrystopher L. Nehaniv and Kerstin Dautenhahn. The Correspondence problem. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*. MIT Press, 2002.

Luke Rendell and Hal Whitehead. Culture in whales and dolphins. *Behavioral and Brain Sciences*, 24(2):309-382, 2001.

Andrew Whiten, Jane Goodall, W.C. McGrew, T. Nishida, V. Reynolds, Y. Sugiyama, C. E. G. Tutin, R. W. Wrangham, and C. Boesch. Cultures in chimpanzees. *Nature*, 399:682-685, 1999.