Solving the Correspondence Problem in Robotic Imitation across Embodiments: Synchrony, Perception, and Culture in Artifacts

Aris Alissandrakis, Chrystopher L. Nehaniv and Kerstin Dautenhahn

1 The 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 [7, 8]. 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 agents individual and learning history etc. [4].

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* [12]:

> Given an observed behaviour of the model, which from a given starting state leads the model through a sequence (or hierarchy [or program]) of sub-goals in states, action and/or effects, one must find and execute a sequence of actions using ones 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 (**A**ction **L**earning via **I**mitation between **C**orresponding **E**mbodiments) generic imitation framework, and implemented it in different simple software testbeds[2].

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 is (at least partially) solved.

To address this in an easy to generalize way, we developed ALICE (**A**ction **L**earning for **I**mitation via **C**orrespondences between **E**mbodiments) as a generic framework for building up correspondences based on *any* generating method for attempts at imitation. This framework is related to statistical string parsing models of social learning from ethology [3] and also the Associative Sequence Learning (ASL) theory from psychology [6].

The ALICE framework (shown in Fig. 1) 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 imitators resulting behaviour [1], and on whether it should be characterized as 'imitation', 'emulation', 'goal emulation', etc. [12].

[1] For a formal statement of the correspondence problem relating to the use of different error metrics and for other applications, see also [9, 10, 11]

[2] These testbeds were implemented using the Swarm agent simulation system (`http:\\wiki.swarm.org`).

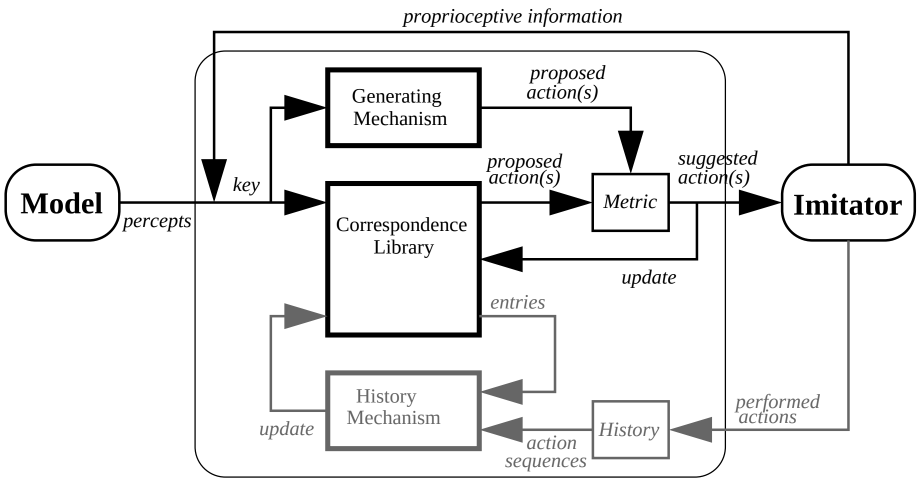


Fig. 1: **The ALICE framework.** The *percepts* of the imitator arising from the model's behavior (actions, states and effects) and *proprioceptive information* (state) of the imitator form a *key* that is used by the *correspondence library* (if it matches any of the existing entry keys at that stage of the library's growth) and the *generating mechanism* to produce a sequence of one or more *proposed action(s)*. These are evaluated using a *metric*, and the correspondence library is updated accordingly with the resulting *suggested action(s)* for the imitator. In parallel (shown in the figure using a gray color), the *history mechanism* can be used to discover any *action sequences* from the *history*, that can improve any of the existing library entries. The history is composed by the sequence of all the actions performed so far by the imitator.

2.1 The Generating Mechanism

The *generating mechanism* is used in the ALICE framework to produce the contents of the correspondence library; it can be *any* algorithm or mechanism that generates actions (or sequences of actions) that are valid (i.e. within the agent's repertoire) and possible in the context of the imitator. 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. In the ALICE framework, no direct feedback from the model is used, instead the metrics are used to evaluate the imitation attempts.

2.2 The History Mechanism

If only the actions found by the generating mechanism are used to build-up the correspondence library, the performance of the imitator would be directly limited by the choice of the algorithm. Moreover, some of the stored actions, although valid solutions to the correspondence problem, may become invalid in certain contexts. The *history mechanism* helps to overcome these difficulties: The imitator can examine its own history to discover further correspondences without having to modify or improve the generating algorithm used. These correspondences will be *sequences of* actions since, no matter how simplistic, the generating mechanism is required to be able to explore the entire search-space of single actions. An agent's *history* is defined as the list of actions that were performed so far by the agent while imitating the model together with their resulting state and effects. This kind of history provides valuable experience data that can then be used to extract useful mappings to improve and add to the correspondence library created up to that point. This approach can be useful to overcome possible limitations of the generating mechanism [1].

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[3].

If instead an entry already exists, the new action is compared to the stored action[4]. 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 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.

[3] More precisely, the contents of the perceptual key depend on the metric the agent is using, for example each of the keys will only contain state(s) and action(s) if a composite state-action metric is used.

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

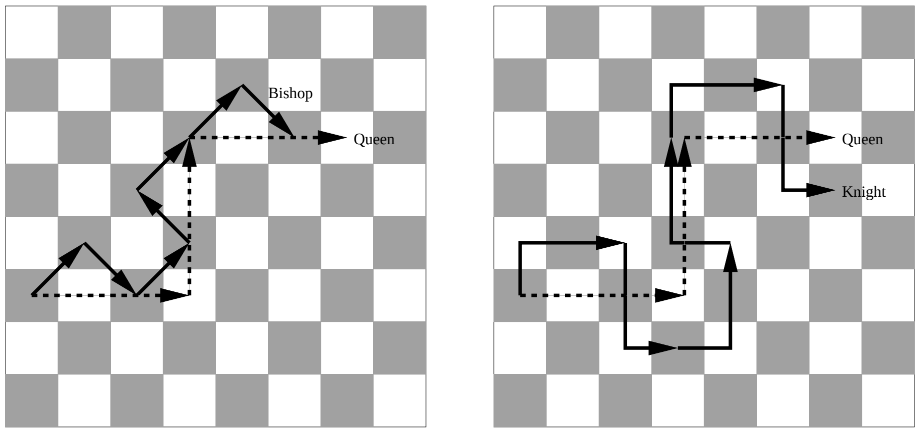


Fig. 2: **Two Chessworld examples.** Two imitator agents (solid paths), a Bishop (left) and a Knight (right) attempt to imitate the movements of the model Queen agent (dotted path).

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.

A more detailed description of the ALICE framework can be found in [2].

3 The Chessworld Testbed

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.

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 displacements achieved and then tries to imitate them, 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 a 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

The ALICE realization in Chessworld (seen in Fig. 3) 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 simple greedy algorithm, returning sequences of actions from the imitator agent's repertoire. 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 testbed, see [1].

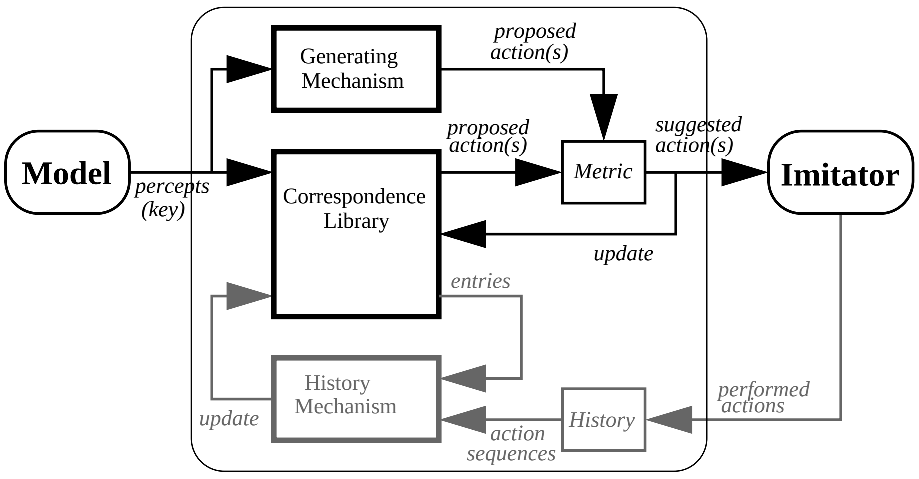


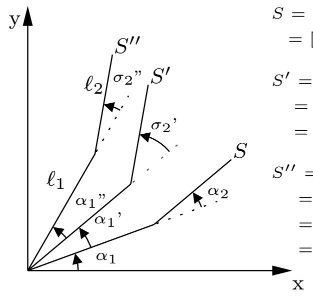
Fig. 3: **The ALICE framework as realized in the Chessworld testbed.** Note that compared to the generic version of the framework (shown in Figure 1), *proprioceptive information* from the imitator is not used here, as the realization of the ALICE framework in Chessworld only considers the *action* aspects of the agent's behavior and not the *state* or the *effects.* Other components are as explained in Fig. 1.

4 The Rabit Testbed

The Rabit (**R**obotic **A**rm em**B**odiment for **I**mitation **T**estbed) environment was created as simple, yet 'rich enough' to allow for several dissimilarly embodied model and imitator agents to be considered. A Rabit agent (see Fig. 4) occupies a two-dimensional workspace and is embodied as a robotic arm that can have any number of rotary joints, each of varying length. Each agent embodiment is described by the vector $L = [\ell_1, \ell_2, \ell_3 \cdots, \ell_n]$, 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, $A = [\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n]$, where n is the number of its 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, $S = [\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n]$, where n is the number of its joints. A distinction can be made 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



State

$S = [\sigma_1,\, \sigma_2]$
$\quad = [\alpha_1,\, \alpha_2]$

$S' = [\sigma_1',\, \sigma_2']$
$\quad = [(\sigma_1{+}\alpha_1'),\, (\sigma_2{+}\alpha_2')]$
$\quad = [(\alpha_1{+}\alpha_1'),\, (\alpha_2{+}\alpha_2')]$

$S'' = [\sigma_1'',\, \sigma_2'']$
$\quad = [(\sigma_1'{+}\alpha_1''),\, (\sigma_2'{+}\alpha_2'')]$
$\quad = [(\sigma_1{+}\alpha_1{+}\alpha_1'),\, (\sigma_2{+}\alpha_2{+}\alpha_2')]$
$\quad = [(\alpha_1{+}\alpha_1'{+}\alpha_1''),\, (\alpha_2{+}\alpha_2'{+}\alpha_2'')]$

Action

$A = [\alpha_1,\, \alpha_2]$
$A' = [\alpha_1',\, \alpha_2']$
$A'' = [\alpha_1'',\, \alpha_2'']$

Embodiment

$L = [\ell_1,\, \ell_2]$

Fig. 4: **Example embodiment of a Rabit agent.** A two-joint robotic arm, with arms of length ℓ_1 and ℓ_2, moving from 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.

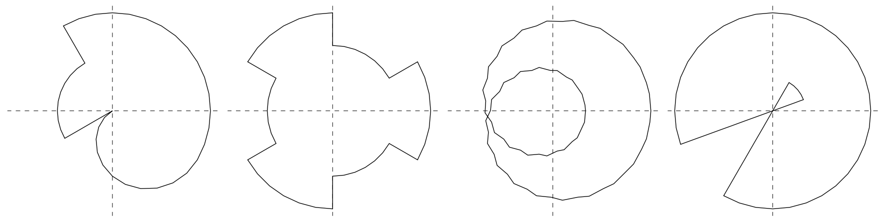


Fig. 5: **Different examples of Rabit behaviours.** Shown are four different effect trails (the agent embodiment is not shown), drawn by the end tip of the each agent manipulator arm. All agents shown have the same embodiment $L = [20, 20, 20]$.

360°) in but only in multiples of 10°.

The end tip of the arm can leave a trail of 'paint' on the workspace, 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 internally 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 behaviour 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 testbed with the fixed base rotary robotic arms favours circular looping effects and the model behaviours used in the experiments were designed as such (see Fig. 5).

Each complete behaviour (or "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. An example of using the different metrics described below is shown in Fig. 6.

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[5] (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.

$$x_i = \sum_{j=1}^{i-1} x_j + \ell_i \cos(\sum_{j=1}^{i} \sigma_j) \tag{1a}$$

$$y_i = \sum_{j=1}^{i-1} y_j + \ell_i \sin(\sum_{j=1}^{i} \sigma_j) \tag{1b}$$

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.

$$d_i = \sqrt{(x_i^{\text{model}} - x_i^{\text{imitator}})^2 + (y_i^{\text{model}} - y_i^{\text{imitator}})^2} \tag{2}$$

$$\mu^{state} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{3}$$

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 $(ratio \times i)^{\text{th}}$ 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.

[5] 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.

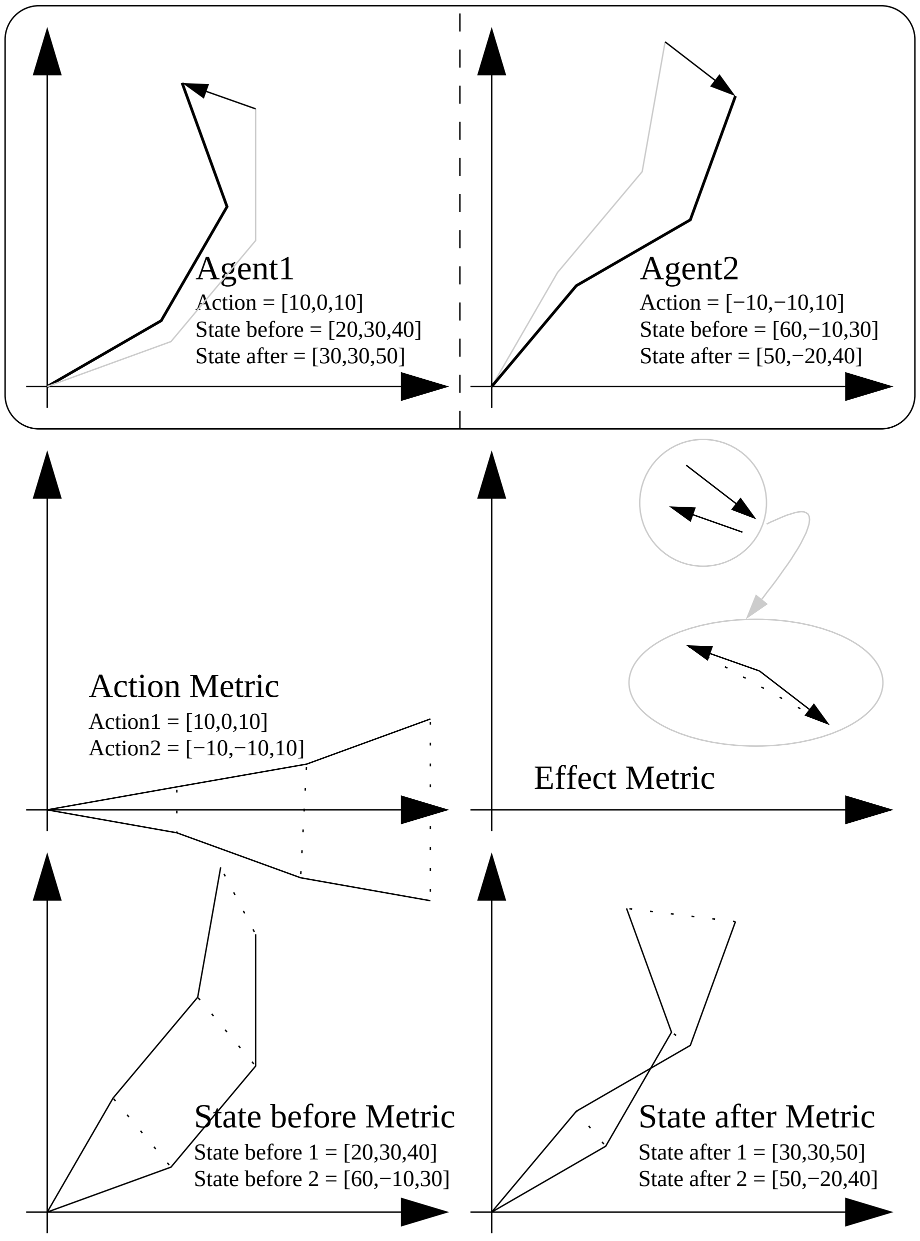

Agent1
Action = [10,0,10]
State before = [20,30,40]
State after = [30,30,50]
Agent2
Action = [−10,−10,10]
State before = [60,−10,30]
State after = [50,−20,40]
Action Metric
Action1 = [10,0,10]
Action2 = [−10,−10,10]
Effect Metric
State before Metric
State before 1 = [20,30,40]
State before 2 = [60,−10,30]
State after Metric
State after 1 = [30,30,50]
State after 2 = [50,−20,40]

Fig. 6: **An example of using the metrics to compare actions, states (before and after) and effects between two Rabit agents, *Agent1* (top, left) and *Agent2* (top, right).** The figure visualizes the vectors used (depending on the metric) and the distances that are summed and then averaged to give each metric value. Both agents have the same embodiment $L = [20, 20, 20]$.

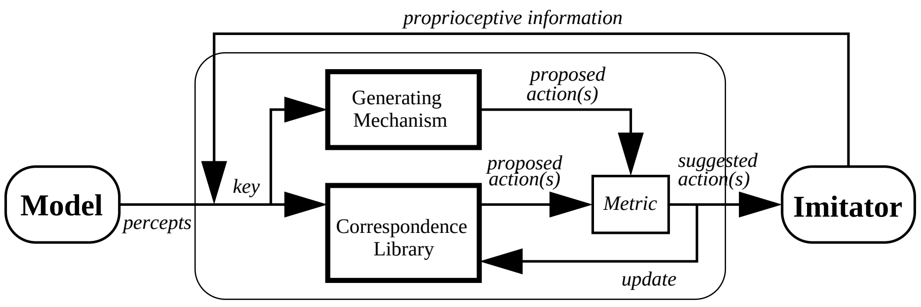


Fig. 7: **The ALICE framework as realized in the Rabit testbed.** Note that compared to the generic version of the framework (shown in Figure 1), the *history mechanism* is not used in parallel to update the correspondence library, as the realization of the ALICE framework in Rabit testbed considers only single actions and not sequences. Other components are as explained in Fig. 1.

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 of the vector difference between two effects (x_1, y_1) and (x_2, y_2).

$$\mu^{effect} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4}$$

4.2 ALICE implementation

In the Rabit implementation of ALICE, each entry in the correspondence library can use as a key the action/state/effect of the observed model agent and the current state of the imitator, as perceptual and proprioceptive components respectively. The key can be composed of just a single of these aspects (e.g. action only), or a combination (e.g. action, state and the imitator's state for proprioception).

For the generating mechanism, an algorithm that returns single random (yet valid) actions is used. It is possible to replace it 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.

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", controlled by a threshold. We call this *loose perceptual matching* and we hypothesized that it should support learning and generalization.

In the current implementation, each entry can store up to three possible corresponding actions that can be seen as possible alternatives[6].

For a more detailed description of Rabit and the ALICE implementation in this testbed, see [2].

5 Experiments on aspects of imitation

Using the robotic arm testbed we conducted various experiments to study the possibility of social transmission of behaviours through heterogeneous agents, the effect 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 and emergence of 'proto-culture'

Besides being a powerful learning mechanism, imitation broadly construed is required for cultural transmission (e.g. [5]). Transmission of behavioural skills by social learning mechanisms like imitation may also be fundamental in non-human cultures, e.g. in chimpanzees [14], whales and dolphins [13]. The robotic arm testbed 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. 8 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 is transmitted perfectly. This is 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.

[6] Note that the history mechanism which also considers sequences of past imitative attempts when updating the correspondence library entries is not implemented in the Rabit testbed since simple action to action correspondence suffices here. In contrast, corresponding 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.

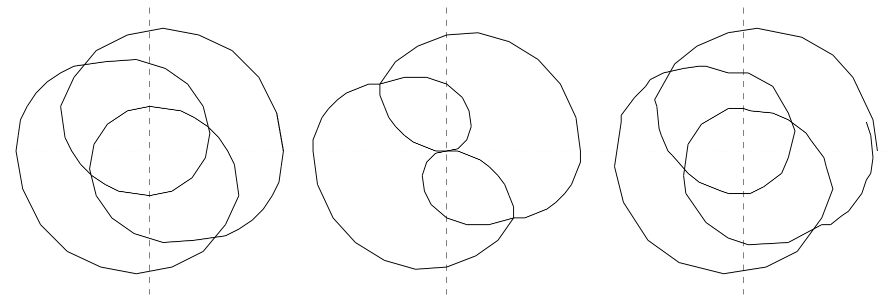


Fig. 8: **An example of social transmission.** The original model ($L = [20, 20, 20]$) is shown to the left. In the middle, an imitator ($L = [30, 30]$) acts also as a model for the imitator on the right ($L = [20, 20, 20]$). Both imitators use the action metric.

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. 8, 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.

The examples shown in Figs. 9 and 10 illustrate the emergence of 'protoculture' in a cyclically ordered chain of three and eight imitators with no overall model. The agents imitate only the agent clockwise from them, using the action metric. Initially they move randomly, as the generating mechanism is trying to discover correspondences for the (also random) actions of their model. Over time, they are able to imitate each other's actions and a stable behavioral pattern emerges.

Different runs yield different emergent culturally sustained behaviors. The location and orientation of the emergent pattern is different in each agent's workspace, since the location and orientation are irrelevant to the action metric; they will depend on the state of the agent at the moment that it has solved its correspondence problem. Each agent's state will vary as a result of the agents not synchronizing.

The cultural transmission of skills through a heterogeneous population of robots using the ALICE framework could potentially be applied to the acquisition and transmission of skills in more complex populations of robots, involved in carrying out useful tasks, e.g. on the shop-floor of a factory, with new robots coming and going acquiring behaviors by observation without having to be explicitly programmed and without humans having to develop different control programs for different types of robots that need to perform the same task. Instead, the robots would autonomously create their own programs (using social learning) and correspondence libraries, even as new types of robots with

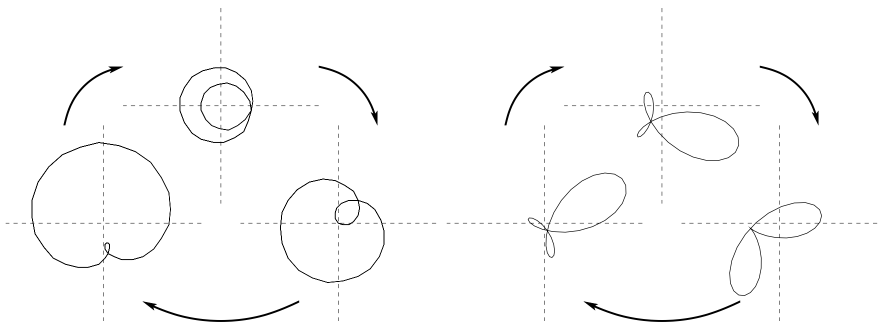


Fig. 9: **Examples of emerging 'proto-culture'.** Two groups of three Rabit agents. All the agents on the left have $L = [20, 20, 20]$ and all the agents on the right have $L = [15, 15, 15, 15]$. Each agent imitates the agent on its left (anti-clockwise) and acts as a model for the agent on its right (clockwise). The emergent behaviour is composed of a single action. $[10, 10, 0]$ left, $[10, -10, -10, -10]$ right.

different embodiments come and go from the population.

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 (see section 5.4 below). 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. 11, 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

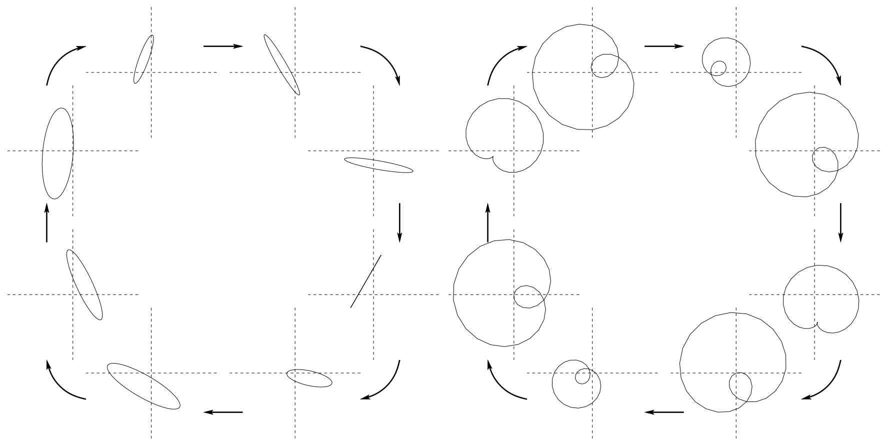


Fig. 10: **More examples of emerging 'proto-culture'.** Two groups of eight Rabit agents. All the agents on the left have $L = [15, 15, 15, 15]$. The agents on the right have alternating embodiments $L = [30, 30]$ and $L = [15, 15, 15, 15]$. Each agent imitates the agent on its left (anti-clockwise) and acts as a model for the agent on its right (clockwise). The emergent behaviour is composed of a single action. $[10, -10, -10, 0]$ left, $[10, 10]$ and $[0, 10, 0, 10]$ right

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 imitators 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.

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. 12 (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

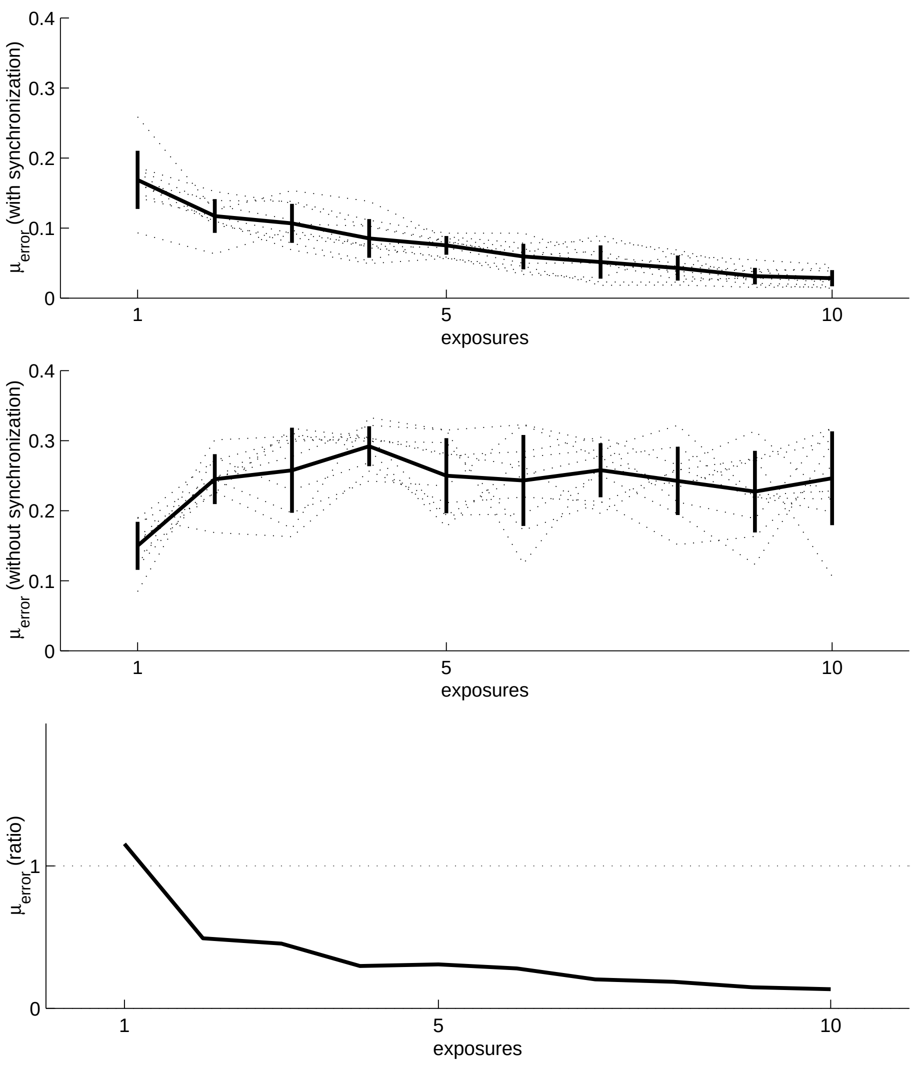


Fig. 11: **Experiments comparing the use of synchronization.** The average maximum error metric value of robotic agents over 10 exposures using synchronization (top panel) vs. not using synchronization (middle panel). The ratio of the maximum error per exposure of the imitating agent using synchronization 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.

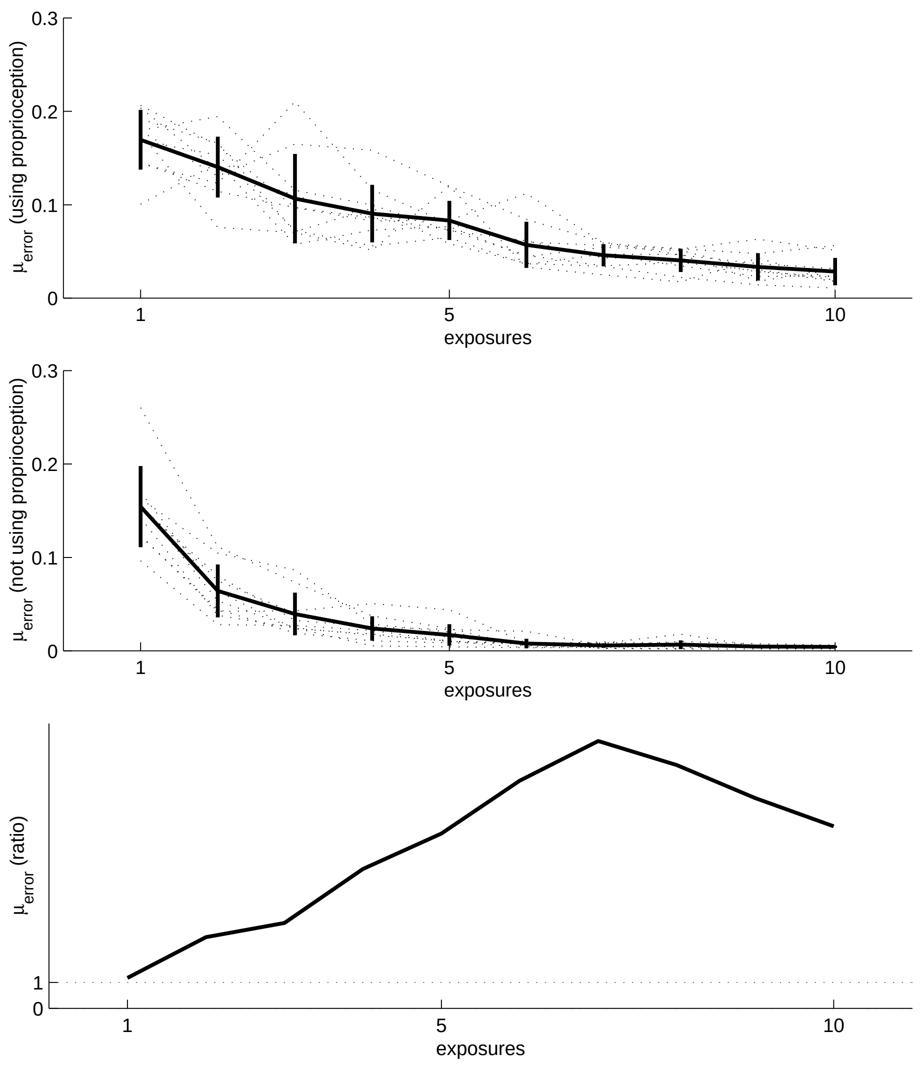


Fig. 12: **Experiments comparing using and not using proprioception.** The maximum error metric value of robotic agents over 10 exposures not using proprioception (top panel) vs. using proprioception (middle panel) when searching through the correspondence library entry keys. The ratio of the maximum 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.

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[7].

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. 13, 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. 13, 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. 13, 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.

5.5 Changes in the agent embodiment

For each agent, vector L defines its embodiment, the number of arm segments and their lengths. We can 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

[7] 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.

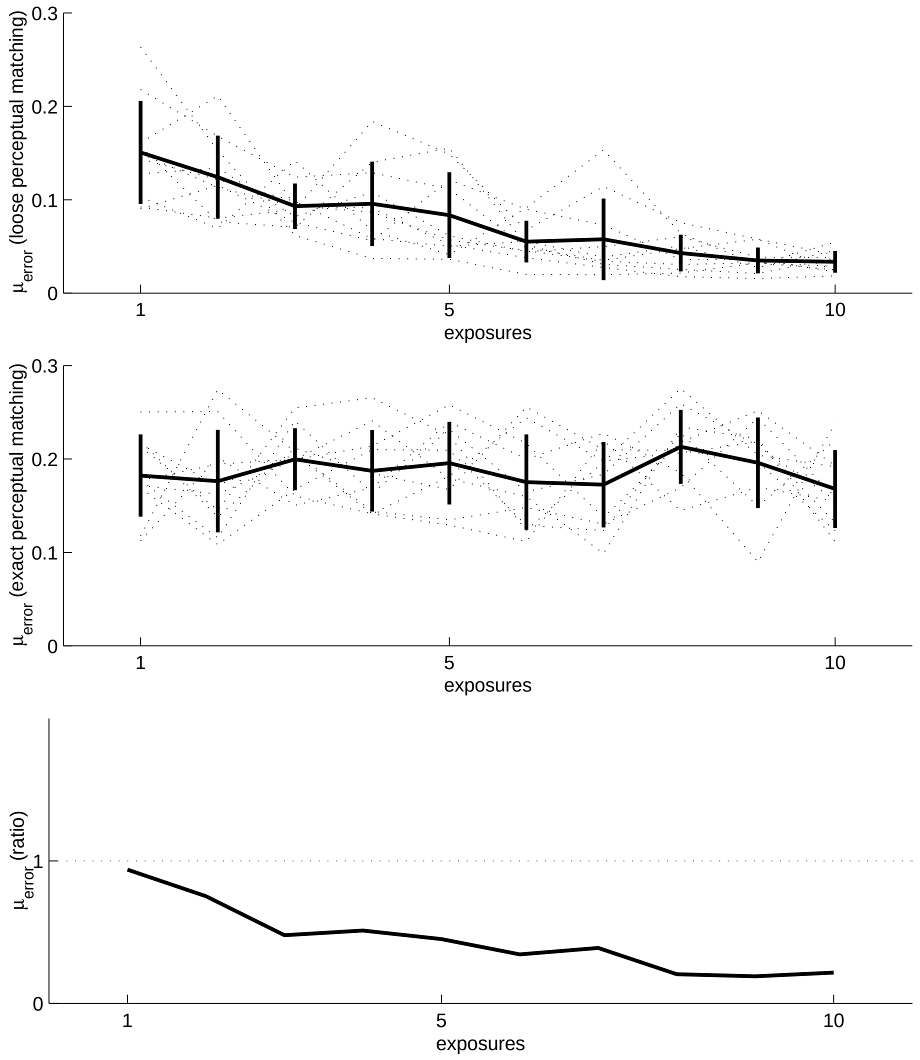


Fig. 13: **Experiments comparing the use of loose perceptual matching.** The average maximum error metric value of robotic agents over 10 exposures using loose matching (top panel) vs. using exact matching (middle panel). The ratio of the maximum error per exposure of the imitating agent using loose 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.

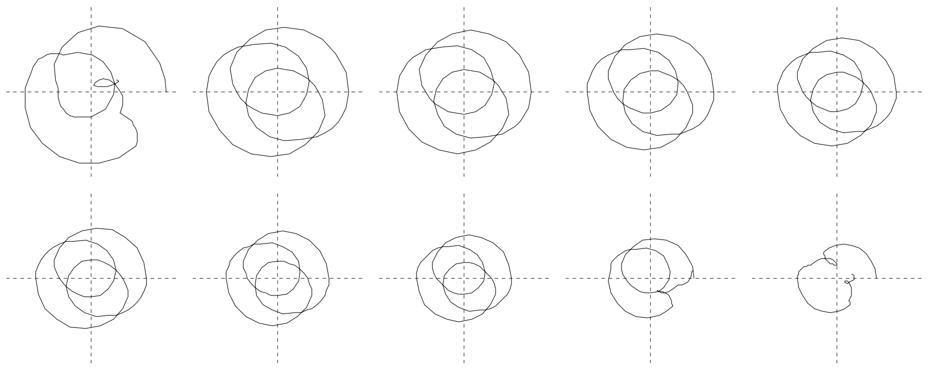


Fig. 14: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [20, 20, 20]$ (top left). Shown are the effect trails of the imitator (left to right, top to bottom) for ten consecutive imitation attempts. A growth vector $G = [-1, -1, -1]$ is used.The final embodiment is $L = [10, 10, 10]$ (bottom right).

must remain constant because such a change makes the existing contents so far of the correspondence library invalid[8]. The growth vector can be used to simulate the body development of the imitator agent during the learning process.

Figs. 14 to 16 show examples of imitator agents that try to imitate a model while a growth vector is used after each imitation attempt, modifying their embodiment[9]. In these examples, the growth vecors equally expand or shorten the length of the imitator's arm segments. Although the imitator constantly changes embodiment, the learning process is relatively unaffected, resulting in a robust imitation performance.

The metric used in these examples is the action metric, compensating for the large range of dissimilar embodiments, and the difference in what they afford. 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. Fig. 17 shows an example of the qualitative effect if the state metric is used, instead of the action metric. The growth vector used is $G = [0, -1, 0]$, shortening the imitator's middle arm segment. The action metric is less affected by the embodiment modification, resulting in a "smaller" version of the model's effect trail (shown in gray). In contrast, the imitator using the state metric effectively

[8] 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.

[9] The model agents preserve a constant embodiment.

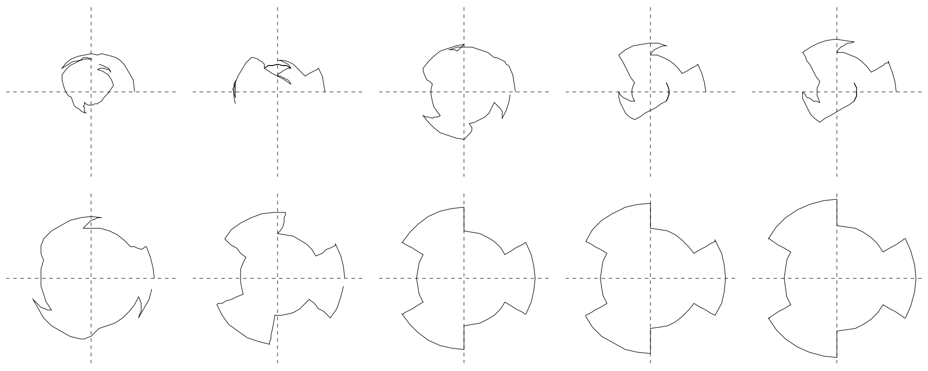


Fig. 15: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [10, 10, 10]$ (top left). A growth vector $G = [1, 1, 1]$ is used.The final embodiment is $L = [20, 20, 20]$ (bottom right).

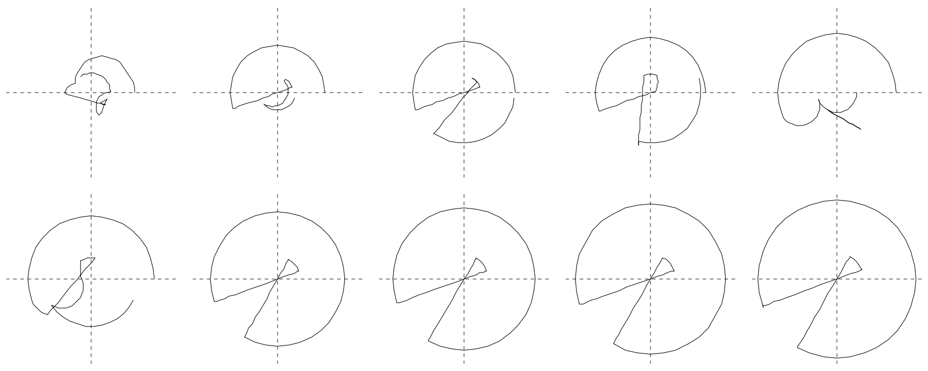


Fig. 16: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [10, 10, 10]$ (top left). A growth vector $G = [1, 1, 1]$ is used.The final embodiment is $L = [20, 20, 20]$ (bottom right).

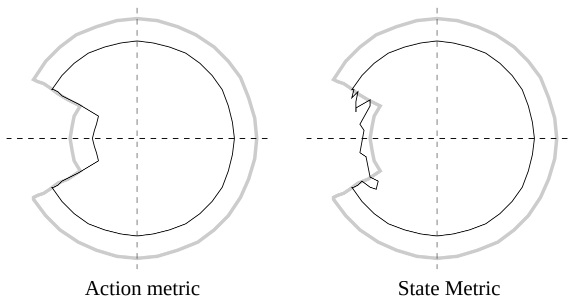


Fig. 17: **Qualitative effect of the metric used by an imitating agent that changes embodiment.** Both imitator agents have the same initial embodiment as the model ($L = [20, 20, 20]$). The figure showns the imitators after ten imitation attempts, having used a growth vector $G = [0, -1, 0]$ after each exposure, with modified embodiments $L = [20, 10, 20]$. The imitator on the left used the action metric, while the imitator on the right used the state metric. The superimposed grey trail shows the model effect pattern for qualitative comparison.

tries to conserve the shape of the pattern by performing actions that acheive similar states.

These examples show that the ALICE mechanism can be robust enough (with a certain tolerance) to compensate for embodiment changes during the initial learning stage (or even later, if the imitator can be again exposed to the model).

6 Conclusions and Discussion

In nature, many organisms' bodies grow and change in the course of their lives. Still the ones that learn socially are able to retain and adapt socially transmitted capabilities despite these changes, whether injurious or natural, to their embodiment. Robots too and other artificial agents that learn socially could benefit from such robustness to embodiment changes. Such a capacity to adapt socially learning despite embodiment change has been demonstrated here via an artificial intelligence learning mechanism framework (ALICE), where the learning is guided by previous experience and evaluation according to given metrics to solve a correspondence problem.

We also showed that loose perceptual matching and synchronization with the demonstrator each resulted in faster learning with lower error rates. Counterintuitively, in the experiments here use of proprioception in building up a correspondence slowed learning. This is most likely due to the larger state space — there is more to learn if proprioception is employed; however, we hypothesize that further work will show that its employment is ultimately beneficial for longer term learn in more complex scenarios.

The work here demonstrates the principle that artificial social learning mechanisms, such as implementations of ALICE, for solving the correspondence problem can be used in populations of robots or agents, to achieve cultural transmission in such populations, even heterogeneous ones consisting of individuals whose embodiments are dissimilar. This may in the future prove useful in the autonomous social learning and adaptation of groups of robots, on factory shop floor or elsewhere, and in social learning interactions in which heterogeneous agents are in involved, e.g. in human-robot interaction. For example, a human might demonstrate a task to a factory robot, which then carries it out, adapting its actions even when its embodiment is perturbed (e.g. by wear-and-tear). Later, when new model robots with different kinds of actuators, degrees of freedom, and so on, come to work in the factory, they acquire skills and task capabilities by learning socially from the robots that are already there. Over generations of robots, cultural knowledge is transmitted, with the robots adapting it to their own changing embodiments.

Acknowledgments

The work described in this paper was partially conducted within the EU Integrated COGNIRON ("The Cognitive Robot Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

References

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Trans. Systems, Man & Cybernetics: Part A*, 32(4):482–496, 2002.

[2] Aris Alissandrakis. *Imitation and Solving the Correspondence Problem for Dissimilar Embodiments – A Generic Framework*. PhD thesis, University of Hertfordshire, 2003.

[3] R. W. Byrne. Imitation without intentionality. using string parsing to copy the organization of behaviour. *Animal Cognition*, 2:63–72, 1999.

[4] K. Dautenhahn and C. L. Nehaniv. An agent-based perspective on imitation. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 1–40. MIT Press, 2002.

[5] Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1976.

[6] C. M. Heyes and E. D. Ray. What is the signtificance of imitation in animals? *Advances in the Study of Behavior*, 29:215–245, 2000.

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

[8] 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 International Workshop on Intelligent Robots and Systems IROS '90*, pages 567–574, 1990.

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

[10] C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Interdisciplinary Approaches to Robot Learning*, pages 136–161. World Scientific Series in Robotics and Intelligent Systems, 2000.

[11] C. L. Nehaniv and K. Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11–51, 2001.

[12] C. L. Nehaniv and K. Dautenhahn. The correspondence problem. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press, 2002.

[13] L. Rendell and H. Whitehead. Culture in whales and dolphins. *Behavioral and Brain Sciences*, 24(2):309–382, 2001.

[14] A. Whiten, J. 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.