Movement Primitives as a Robotic Tool to Interpret Trajectories through Learning-by-doing

Andrea Soltoggio[1] Andre Lemme[1]

[1]Research Institute for Cognition and Robotics, Bielefeld University, 33615 Bielefeld, Germany

Preprint accepted for publication in the International Journal of Automation and Computing (2013)

Abstract: Articulated movements are fundamental in many human and robotic tasks. While humans can learn and generalise arbitrarily long sequences of movements, and particularly can optimise them to fit the constraints and features of their body, robots are often programmed to execute point-to-point precise but fixed patterns. This study proposes a new approach to interpreting and reproducing articulated and complex trajectories as a set of known robot-based primitives. Instead of achieving accurate reproductions, the proposed approach aims at interpreting data in an agent-centred fashion, according to an agent's primitive movements. The method improves the accuracy of a reproduction with an incremental process that seeks first a rough approximation by capturing the most essential features of a demonstrated trajectory. Observing the discrepancy between the demonstrated and reproduced trajectories, the process then proceeds with incremental decompositions and new searches in sub-optimal parts of the trajectory. The aim is to achieve an agent-centred interpretation and progressive learning that fits in the first place the robots' capability, as opposed to a data-centred decomposition analysis. Tests on both geometric and human generated trajectories reveal that the use of own primitives results in remarkable robustness and generalisation properties of the method. In particular, because trajectories are understood and abstracted by means of agent-optimised primitives, the method has two main features: (1) reproduced trajectories are general and represent an abstraction of the data, and (2) the algorithm is capable of reconstructing highly noisy or corrupted data without pre-processing thanks to an implicit and emergent noise suppression and feature detection. This study suggests a novel bio-inspired approach to interpreting, learning and reproducing articulated movements and trajectories. Possible applications include drawing, writing, movement generation, object manipulation and other tasks where the performance requires human-like interpretation and generalisation capabilities.

Keywords: Movement primitives, Learning, Pattern matching, Trajectory decomposition.

1 Introduction

Humans and animals are capable of learning, perfecting and reproducing complex trajectories that allow them to perform a variety of tasks, from coordinated body movements to catching, and particularly in humans, object manipulation, writing and drawing. The mechanisms underlying motor skills, from the learning of basic primitives to their organisation in higher-level cognitive structures, are fundamental in understanding how humans accomplish advanced motor skills [1]. Object manipulation, skilful movements and generalised trajectories are considered fundamental in the evolution of intelligence and modern technology [2]. The autonomous learning of robotic movements and their organisation are an increasingly important research focus.

Object manipulation and precise movements are implemented in industrial robots for manufacturing and production processes. However, a considerable limitation in such movements is that trajectories are often pre-programmed and executed point-to-point, therefore lacking a general and symbolic representation of the movement, as well as the capability of adapting and improving.

One solution to point-to-point representations are movement primitives, short movement or strokes that represent elementary building blocks for more complex movements. Motor primitives represent a biological hypothesis on how complex movements are formed in human and animals [3, 4, 5]. One intrinsic feature of motor primitives is that they generate basic and general movements that can then be combined to compose arbitrary long and convoluted trajectories. While biological studies continue to reveal the neurological bases of motor primitives [6, 7], computational models provide examples of computer generated primitives. Among those, Dynamic Movement Primitives (DMP) [8, 9], Gaussian Mixture Models (GMM) [10, 11], feedforward neural networks [12] and Recurrent Neural Networks (RNN) [13, 14] have gained considerable attention as mathematical tools to generate simple primitive-like movements.

Fundamental questions that arise with the use of primitives are: what are the features of a set of primitives? How are primitives composed to perform articulated movements? And what role do they play in interpreting, coding and learning complex movements? Some approaches start by analysing the demonstrated trajectory employing polynomial decomposition [15], Hidden Markov Models [16], Non-Negative Matrix Factorisation [17], detection of critical points [18] and Guassian Observation Model [19]. Other methods employ reinforcement learning to refine an approximation over time by means of reward signals [20, 21, 22, 23, 24]. Finally, methods have been proposed to join segments to achieve natural-looking trajectories by blending [25, 26] and co-articulation [27, 28]. Algorithms that combine primitive or shape-identification, trajectory segmentation and on-line learning have also be proposed [29, 18, 24] to integrate various subproblems in more capable learning algorithms.

Most algorithms attempt to learn simultaneously both primitives and their use in the decomposition of long trajectories. In contrast, the present method focuses entirely on

Manuscript received date; revised date
A. Soltoggio and A.Lemme contributed equally to this work.

the decomposition, assuming that primitives are previously learnt and already available to an agent. This separation between learning of primitives and decomposition of demonstrations implies that existing and established methods for learning and executing primitives (e.g. Dynamic Movement Primitives (DMP) [8, 9], Stable Estimator of Dynamical Systems (SEDS) [11], Extreme Learning Machine [12, 30]) can be employed in combination with the current algorithm. In other words, any set of primitives can be chosen and used in combination with the proposed decomposition method. This ensures that existing robotic platforms and primitives can be enhanced with the current algorithm to decompose and reproduce articulated trajectories.

The agent-centred approach [31] implies that, as opposed to other approaches [16, 17, 18], the algorithm does not analyse directly the demonstrated trajectory to find features, e.g. inflection points, points of discontinuous derivative, critical points, etc. The demonstrated trajectory is approximated instead by means of a process of learning-by-doing in which the performance, or the accuracy of a reproduction, is improved over time with increasingly refined decompositions. Avoiding the analysis of demonstrated data results in two main features of the algorithm. The first is that a reproduction of a demonstration is biased by the agent's set of primitives. In this respect, the reproduction represents an *interpretation* of a demonstration. In other words, any demonstration, which was generated by an unknown process, is being fitted with the agent's fixed primitives. While this may appear as a limitation, it also means that no assumptions on the demonstrated trajectory are required. The agent attempts to achieve a best approximation with its current primitives used a tool to interpret input data. A second feature is that the algorithm may take as input highly noisy and corrupted data and displays implicit noise suppression and feature detection.

One fundamental aspect of the proposed method is that, similarly to [32], the decomposition initiates as a rough approximation based on one single movement primitive. Interestingly, a complex trajectory with many convoluted parts is not likely to be adequately represented by one single stroke. Yet, by adopting this counter-intuitive approach, a fundamental step in an iterative process can be achieved towards further and more precise decompositions. Points of decomposition are progressively discovered during the iterative process. At each iteration, the part of the reproduction with the maximum discrepancy with the demonstrated trajectory is considered for improvement. Thus, segmentation points are introduced with the simple but effective heuristic of observing the point of maximum error. Decomposition points can also be later suppressed if more general primitives are discovered to fit a part of the demonstration. The deletion of segmentation points is a bio-inspired search that, once some main features of a demonstration are captured, it relaxes constraints to find better solutions and overcome local optima. Such an approach is inspired by the early/later practice phases in motor learning [3]. Finally, combining primitives as symbolic entities supports biological theories on the construction of motor skills as mental representations of existing building blocks [1].

It is important to note that the present algorithm only focuses on geometrical properties of the trajectories, while is agnostic to the velocity profiles. This apparent limitation in reality allows for a more flexible interpretation of trajectories, which may not be necessarily determined by the velocity profile used during generation. Once more, no assumption on the demonstration implies that any demonstration can be observed, decomposed and reproduced with the proposed algorithm.

The algorithm is an extension of that proposed in [33]. In the version in this paper, the algorithm uses one additional set of primitives learnt from human data. This test is essential to confirm the claim that the algorithm can perform well with very diverse sets of primitives. Various criteria to compare trajectories are proposed to underline that trajectory matching may vary according to specific domains and requirements. Additionally, tests are extended particularly to assess the capability of reconstructing noisy data, and the performance is verified on an extended set of exemplary trajectories.

The proposed decomposition algorithm lends itself to promising extensions including learning trajectories from multiple examples, hand-writing recognition, decomposition of complex movement patterns for manipulation and combination of skills. The method is tested only in simulation. Tests on robotic platforms, e.g. the iCub humanoid robot [34] or the KUKA lightweight robot arm [35], are promising extensions.

The decomposition algorithm is explained in detail in the next section. Decomposition tests from simple to complex trajectories are shown in Sec. 3. The implications and possible extensions are then discussed in Sec. 4 before the conclusion in Sec. 5.

2 Search, decomposition and interpretation of trajectories

This section explains the algorithm in all its parts, motivates the bio-inspired approach and illustrates all the steps to reproduce the method.

2.1 Sets of primitives as decomposition tools

The decomposition algorithm requires a set of pre-learnt primitives that can be freely chosen and generated by means of a variety of methods. This feature is particularly important to integrate the proposed algorithm with the well established methods for primitive generations cited above. The performance of the algorithm in the decomposition varies in accuracy and approximation according to the set of primitives, as later tests show. Nevertheless, the method can decompose even with very poor sets of primitives.

To show the possible use of different sets of primitives, the current study considers three sets: two sets generated with the Minimum-Jerk Model (MJM) [36] and one with a feedforward network (Extreme Learning Machine) [12, 30] that was trained to reproduce a set of human drawn trajectories. One MJM set is composed of seven symmetric primitives (Fig. 1A), and a more complex set has 51 primitives with symmetric and asymmetric shapes (Fig. 1B). The ELM-set has 6 primitives (Fig. 1C). Details are provided in the Appendix 5. Experiments can be extended to include

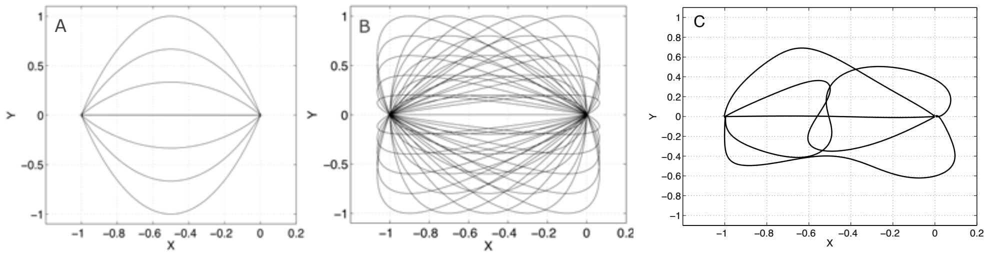

Figure 1 Sets of primitives. (A) This small set of primitives was generated with the Minimum-Jerk Model (MJM) [36] and contains one straight line and six curved lines. It is referred to as the MJM 7-set. (B) This larger set of primitives was generated with the minimum-jerk model and contains 51 primitives both with symmetric and asymmetric geometry (MJM 51-set). (C) This set, composed of only five primitives, was learnt from human demonstrations using an Extreme Learning Machine (ELM) [12], further details on the use of ELM are provided in the Appendix 5.2. Primitives are rotated and scaled to be fitted to the starting and ending point of a demonstration.

further sets.

2.2 Trajectory matching and iterative decomposition

Regardless of the length and complexity of a given trajectory (demonstration), the counter-intuitive position in this study is that only the start and end points are initially considered as extremities of one single primitive. The agent searches among its own primitives a best match. It is assumed that a primitive can be rotated and scaled to connect the initial (I) and final (F) points of the demonstration. If the demonstration is long and articulated, the first match is inevitably a gross approximation. Four numerical criteria to compare trajectories were considered in the current study: (1) a maximum point-wise error (MPE), (2) a point-wise mean square error (PMSE), (3) the area A between the demonstration and the reproduction, and (4) a measure of parallelism Θ. The four criteria are computed by the following equations:

$$\text{MPE} = max(||x(i) - \hat{x}(i)||) \tag{1}$$

$$\text{PMSE} = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(x_j^i - \hat{x}_j^i)^2 \tag{2}$$

$$A = \sum_{i=1}^{N-1} F(x(i), x(i+1), \hat{x}(i), \hat{x}(i+1)) \tag{3}$$

$$\Theta = \frac{1}{N}\sum_{i=1}^{N-1}\frac{x(i+1)-x(i)}{||x(i+1)-x(i)||}\cdot\frac{\hat{x}(i+1)-\hat{x}(i)}{||\hat{x}(i+1)-\hat{x}(i)||} \tag{4}$$

where $x(i)$ is a two dimensional coordinate point of the demonstration, $\hat{x}(i)$ a coordinate point of the reproduced trajectory, M is the dimensionality of the data (2 in the current study), N the number of samples, and F gives the area of the Tetragon specified by the input arguments. The data points $\mathbf{x}$ are obtained by cubic splines interpolation of the original sampling to ensure that their normalised distance and N are equal.

Eqs. 1-4 can be used independently or linearly combined to assess how similar two trajectories are. Eqs. 1-3 are null for perfectly matching trajectories, while Eq. 4 is equal to 1 for matching trajectories. Visual observation over many examples revealed that deriving a measure of similarities between two different trajectories is not immediate. In effect, evaluating similarities between trajectories may be domain-dependent or even subjective. The focus of the study is not to compare the performance of Eqs. 1-4, nor to propose a best criterion. Different matching criteria are proposed here as alternatives which can be chosen to work with the present algorithm. The tests in the current study use by default Eq. 2 because it produced predictable segmentations on a large variety of demonstrations. Eqs. 1 and 3 are also employed in tests to show the robustness of the method.

Once a best matching primitive is identified, the point x^* that returns the maximum error in Eq. 1 on the demonstration D

$$x^* : MPE = max(||x(i) - \hat{x}(i)||), \forall i \in D \tag{5}$$

is chosen as candidate segmentation point. Thus, the first approximation is used to identify a first decomposition point along the demonstration, i.e. a first point to use in an iterative process of further decompositions. Once x^* is identified, each sub-trajectory to the left and to the right is matched with a best primitive. Two cases are now possible: (1) the reproduction with x^* as segmentation point brings an improvement with respect to the matching criteria expressed by Eqs. 1-4; (2) the segmentation does not bring an improvement. In the first case, the *candidate* segmentation point is promoted to *established* segmentation point and the iterative process can continue on each segment. In the second case, the segment is labelled as final and no further segmentation is considered. Fig. 2 illustrates the first four steps of the iterative process on an trajectory.

The heuristic that identifies candidate segmentation points is not based on an optimality measure, which is difficult to infer in an iterative process. The attempt is instead that of identifying potentially appropriate points to improve further a reproduction. The underlying idea is that the furthest point on the demonstration from the current reproduction lays potentially in a part of the demonstration that is not correctly represented by the reproduction, and thus requires further segmentation. This simple heuristic proves effective as demonstrated later in simulations.

2.3 From sequences of points to sequences of primitives

When decomposed finely, any trajectory can be represented as a sequence of close points united by straight lines. A decomposition that reproduced exactly the demonstration in such a way minimises the reproduction error. However, such a decomposition merely copies a demon-

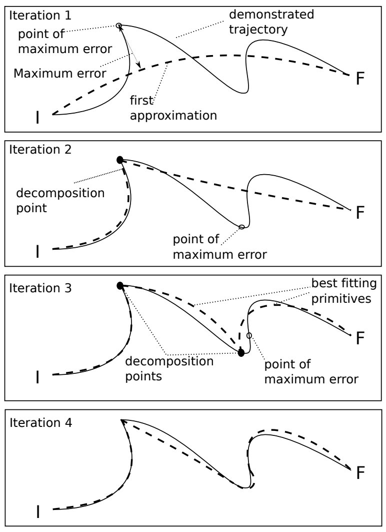

Figure 2 Graphical illustration of the iterative decomposition process. Iteration 1 begins by selecting one single primitive as approximation of the global trajectory between the initial (I) and end (E) points. The primitive is chosen applying one of the criteria expressed by Eqs. 1-4. On the demonstration, a candidate segmentation point is chosen according to Eq. 1. In Iteration 2, two better fitting primitives are identified, as well as a new point of maximum error. Iteration 3 and 4 show further steps of the iteration. The algorithm may continue to improve locally the approximation until stop criteria are satisfied.

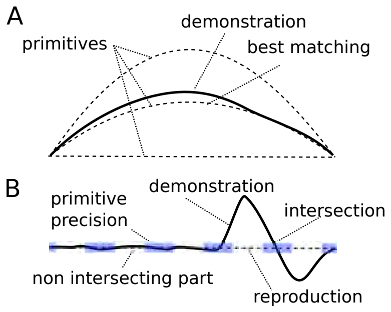

Figure 3 Best fitting and intersections. (A) Three primitives are shown in the attempt to match a demonstrated trajectory. The best matching primitive is not a perfect reproduction, but rather a general abstraction of the demonstration. (B) A demonstration is matched to a straight primitive in which the second dimension, or precision, is shown. The left-most part of the trajectory, although not perfectly straight, does not intersect the two-dimensional primitive, suggesting that no further features are present in the demonstration. The right-most part of the trajectory instead clearly intersects the primitive, indicating the utility of further decompositions.

strated trajectory without generalising the overall shape of a movement. The problem is effectively both a classification problem (finding the best matching primitive) and an optimisation problem (reducing the number of segmentation points). A trade-off between generality, with few decomposition points, and precision, with many segmentation points, is desired and sought [37]. As a rule, generality of one solution is accompanied by a residual error with one particular demonstration. The implication is that decomposing a trajectory to minimise the error may lead to a high number of segmentation points. Most algorithms use a error threshold below which the segmentation is considered satisfactory. This problem derives also form the arguable assumption that trajectories have a length but dimensionless thickness. In robotic and real world scenarios, trajectories are both executed and perceived with a certain tolerance. Accounting for such an aspect is a key aspect to avoid over-fitting, unnecessary computation and excessing segmentation.

The method in this study attempts to mimic a trajectory with given primitives that guarantee generality and may be devised to guarantee also efficiency, optimality, or to conform to particular robotic requirements, without necessary minimising an error measure. For example, the minimum-jerk model used in the current experiments guarantees energy minimisation and is biologically plausible [36], while the ELM-set uses a neural learning paradigm that reproduces human drawn trajectories.

2.3.1 Precision of primitives and intersections

Instead of considering the error between demonstration and reproduction as stopping criterion, the current algorithm looks at whether the demonstration and the reproduction have *intersections*. If they have at least one intersection, the demonstration is assumed to have further features that need decomposing. If there are no intersections, the current primitive is assumed to be the best approximation: further decompositions may reduce the error but also reduce generality.

Intersections are intended as two trajectories crossing each other: however, two noisy and overlapping trajectories have many local intersections that would not be considered such by a human observer. Thus, to detect significant intersections, the algorithm associates a precision value to the primitives. Such a precision is an index of how thin a trajectory may be with respect to its length. In effect, this parameter may encode the precision of a mechanical arm, or may be adjusted to account for the variance of many samples, if those are executed by imprecise human movements. In short, the precision parameter is a necessary element in the interpretation of an observed trajectory. It answers the questions: what are the agent's perception and execution capabilities? What is a realistic precision to be implemented when reproducing a demonstration?

The case is illustrated in Fig. 3 in which the primitive is shown with an associated thickness. If the demonstrated and performed trajectories *do not intersect*, the algorithm infers that there are no further prominent features in the demonstration that need to be reproduced with further segmentations. Non-intersecting trajectories can be somehow distant, but the matching criterion (Eqs. 1-4) ensures that this distance is minimised. No interactions mean effectively that the reproduction and the demonstration are as close as possible given the current set of primitives. One exception is when the demonstration exists the reachable area of all primitives. This is for example the case of a circle drawn with a nearly overlapping start and end point. When the demonstration exists the reachable area of the primitives, further segmentations are enforced.

In the experiments of this paper, the smaller sets (MJM and ELM) have a precision value $p = 20 = 1/0.05$, where 0.05 is the thickness of the primitive normalised to the shortest side of the drawing area. The more accurate 51-primitive set has a precision $p = 100 = 1/0.01$, i.e. the thickness of a primitive is 1% of the drawing area. A thickness of 0 corresponds to infinite precision, a concept that does not describe real data from a demonstration and clearly underlines the importance of considering thickness values higher than 0. Higher precision values can be adopted when the demonstration is known to be very accurate. Intersections are detected analytically by computing the cross products between the direction of the primitive and the error vectors of all points: if the cross products have different signs and the error vectors are greater than the line thickness, then an intersection is detected.

The intersection criterion attempts to capture features of the demonstration that are *observable* with set of primitives used. It is nevertheless possible to use a more traditional stopping criterion, for example requiring that the maximum error is decreased below a certain threshold. Such an approach may be used when more emphasis on minimising the error is necessary and an approximation that respects an error constraint is desired. This variation was experimented in the current algorithm and is easy implementable by letting the algorithm continue the segmentation until the maximum error falls under a threshold. A similar variation may also include, for example, a measure of how parallel two trajectories are (i.e. Eq. 4). The algorithm may be required to continue segmenting until a certain threshold is reached. These variations of the algorithm require more human supervision in setting such an error threshold and understanding what matching criteria are needed in a particular scenario. In some cases, introducing additional matching measurements and stopping criteria may lead to reproductions that are perceived visually as better approximations.

2.3.2 Deleting segmentation points

The iterative process implies that the interpretation of the demonstration (i.e. the solution) varies and improves at each further decomposition. One question is whether decomposition points that were found initially during the process are still good segmentation points later as the accuracy improves. Inspired by theories of motor learning in humans [3], the proposed method introduces a type of search that releases early constraints when a new segment is added. At the insertion of a new decomposition point, primitives are searched to the left and to the right of the candidate point. The search may go beyond the immediate left and right segments. It is possible to search further left and further right, thereby attempting larger generalisations. Neighbouring decomposition points are eliminated if more general primitives without intersections are discovered.

This check guides the search to avoid local optima, and at the same time it helps reduce the number of overall segmentation points, thereby achieving more general solutions. Releasing constraints implies more computation while searching larger primitives that may skip segmentation points. This type of search is nevertheless far from exhaustive: the further exploration relies on the current segmentation. It represents an attempt to reorganise parts of the trajectory

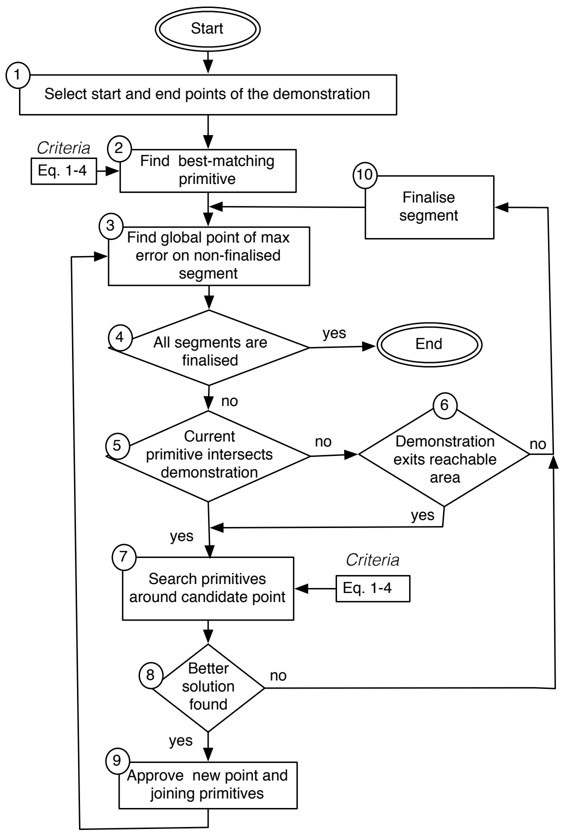
Start
1 Select start and end points of the demonstration
Criteria
Eq. 1-4
2 Find best-matching primitive
10 Finalise segment
3 Find global point of max error on non-finalised segment
4 All segments are finalised
yes
End
no
5 Current primitive intersects demonstration
no
6 Demonstration exits reachable area
no
yes
yes
7 Search primitives around candidate point
Criteria
Eq. 1-4
8 Better solution found
no
yes
9 Approve new point and joining primitives

Figure 4 Flow charts describing the various phases and iterative nature of the algorithm. The numbers that identify each block are used as references in the text to describe each phase.

according to new knowledge that was gathered during the iterative segmentation process.

2.4 The iterative algorithmic procedure

It is now possible to introduce a flow chart to explain in detail the overall procedure. Fig. 4 helps follow the detailed explanation below. The algorithm is also reproducible with the Matlab code provided as support material to this paper and available for download at http://www.cor-lab.de/decomp.

The algorithm starts selecting one primitive that best matches the demonstrated trajectory (Fig. 4, blocks 1 and 2). The best match is obtained comparing all primitives with the demonstration and choosing the primitive that minimises a measure of discrepancy (Eqs. 1 and 3) or maximises a measure of similarity (Eq. 4). In the next step (block 3), the algorithm finds the point of maximum error between the demonstration and the reproduction (Eq. 5). This is a candidate segmentation point and is located in a part of the demonstration that is poorly approximated. Initially there is only one segment. As the iterations proceed, more segments are created. When created, each segment is labelled as *non-finalised*, meaning that further decompositions are possible. The point of maximum error is sought on a non-finalised segment (blocks 3 and 4). The algorithm now checks whether the primitive intersects the demonstration or not (block 5). As illustrated in Figs. 3A and B,

Seg.	Primit.	Start	Scaling	Angle	Final
1	number	xy coor.	factor	angle	y/n
2	..	..	..	..	..

Table 1 Representation of a trajectory as a sequence of primitives. Segments (i.e. rows in the table) are added and occasionally removed during the iterative process. For each segment, it is necessary to specify which primitive is used (2nd column), the starting point (3rd column), the scaling and angle (4th and 5th column) and whether the segment can be further decomposed (6th column).

an intersection suggests the presence of a relevant feature that can be captured with further decompositions. If the best matching primitive does not intersect the demonstration (block 5), the demonstration may be laying outside the reachable area of the primitives (block 6). This case, or the case in which there is an intersection, mean that there are additional features in the demonstration that need to be captured. Therefore, the algorithm proceeds with the segmentation (block 7). Otherwise, the current segment is finalised (block 10). The search in block 7 is carried out by exploring primitives that approximate the left and right parts of the demonstration from the candidate segmentation point. Such a search involves also the elimination of older segmentation points when better approximations are found. The search for better primitives in block 7 may or may not result in an improvement of Eqs. 1-4. If no improvements can be achieved, the segmentation point is rejected and that segment is label as finalised (blocks 8 and 10). If an improvement is found, the segmentation point and the left and right primitives are promoted as part of the current segmentation (block 9).

Throughout the process, the representation of a demonstration is updated. Table 1 shows how the primitive-based symbolic trajectory is described. At the first iteration, only one row is present. Further segmentations add more rows describing primitives, start point, scaling and angle, and whether the segment is finalised.

3 Simulation Results

The current section reports the simulation results of the algorithm applied to a variety of demonstrated trajectories, from simple to complex.

3.1 Reconstructing short demonstrations

The decomposition algorithm is applied here on human and machine generated trajectories affected by noise. These basic examples have the purpose of showing how the algorithm interprets and reconstructs short noisy trajectories. Fig. 5 shows the application of the algorithm to three different demonstrations after they were corrupted with noise. The method favours elegant decompositions with few primitives, resulting in some cases in a residual error between demonstrated and reproduced trajectories. The discrepancy stems from the more general trajectories chosen by the agent with respect to the irregular human generated data. The decomposition with the MJM set of 51 primitives appears more accurate in comparison with the decomposition from the 7-set. The small set of 7 primitives instead

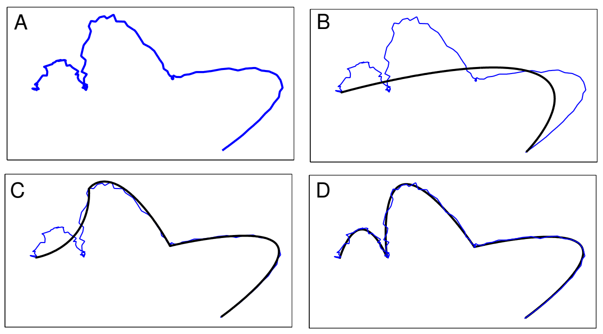

Figure 6 Examples of reconstruction from a noisy machine generate trajectory. (A) The original demonstration affected by noise. (B) The first step of the decomposition. (C) The third step of the decomposition (D) Fourth and final step of the decomposition. The algorithm found the original primitives that were used to draw the demonstration before noise was applied.

captures the main features of the demonstrated trajectories favouring straight lines, effectively achieving a higher level of abstraction. The implication is that in front of complex demonstrations, agents or robots with few primitives can nevertheless utilise the algorithm to decompose a demonstration according to their basic skills. The ELM-set, despite having only five primitives, performed very well. The reason is because the ELM-primitives were trained on the same trajectory later presented for reconstruction. Therefore, the ELM-set contains primitives that match well the demonstrated trajectory. Nevertheless, it must be noted that the demonstrations are not exactly the same as the primitives and, moreover, the data seen by the algorithm is the corrupted data in the second row in Fig. 5.

From this first test it emerges one important and bio-inspired feature of the algorithm. The method appears to reconstruct, in a way to recognise, those trajectories that are similar to the known primitives. The reconstruction by the MJM 51-set in the third row is more abstract and less similar to the original than the reconstruction by the ELM-set, despite the considerably larger library of primitives in the 51-set. However, while the ELM-set performed well in this particular test, the 51-set is more generic and is likely to perform better on other trajectories with arbitrary geometry.

Further tests were performed on automatically generated trajectories with additional high level of noise. Fig. 6 illustrates the capability of the algorithm in reconstructing corrupted data. The demonstration was created with the same primitive set used for the reconstruction, which in part explains the correct matching. Nevertheless, this approach appears biologically plausible because humans too tend to recognise in imprecise images objects and shapes that were previously learnt [38]. As hypothesised also in [39], the reconstruction and reproduction are closely coupled: the present algorithm shows that noisy data are *recognised* with or *fitted* to the known primitives.

The primitives are executed sequentially without additional procedure to join them. Therefore, points of discontinuous derivative are noticeable where primitives

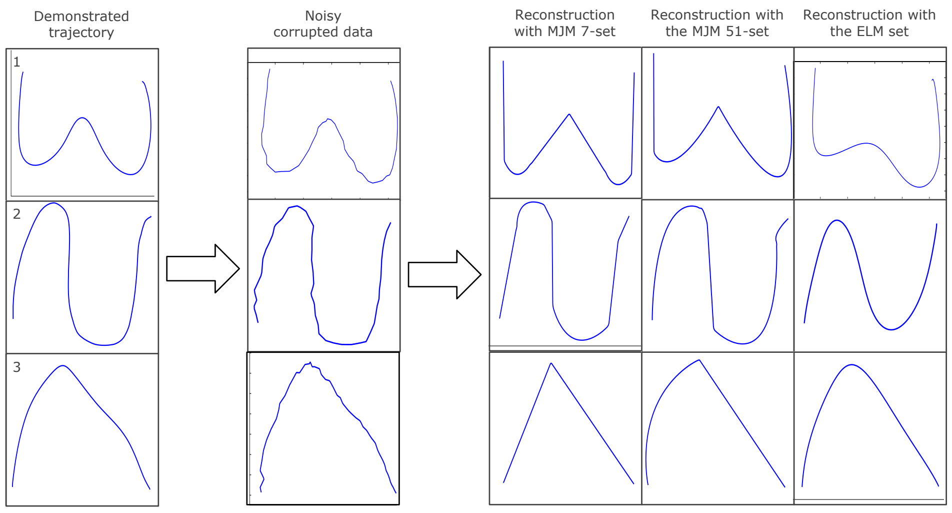

Figure 5 Examples of decompositions of short demonstrations. (First row) A hand-written W (first column) is first corrupted with noise (second column). The noisy data is used to run the algorithm with the 7-set (third row) and with the 51-set (fourth row). Finally, the fifth column shows the interpretation and decomposition with the ELM-set of primitives. The rows below show similar plots for different demonstrations.

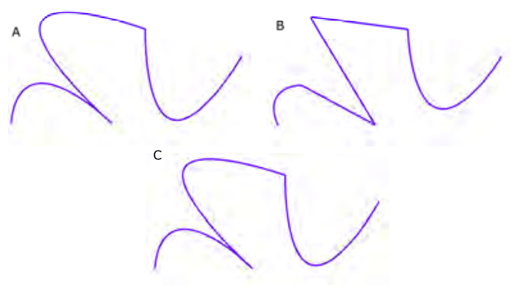

Figure 7 Testing the decomposition on machine generated trajectories. (A) One hundred machine-generated trajectories as this one in the example are used for the extensive performance test. (B) Reproduced trajectory with the small 7-primitive sets. (C) Reproduced trajectory with the large 51-primitive set.

join. Smoothing a trajectory requires the understanding of whether a point is a cuspid, i.e. the trajectory has a discontinuous derivative, or it can be rounded with a co-articulation algorithm [28, 26]. As this particular problem was not the focus of the present algorithm, all primitives are joined without blending or coarticulation.

A further test was executed to assess the performance of the algorithm on different trajectories without noise. One hundred trajectories were constructed with sequences of three primitives from the 51-primitive set, applying random primitive lengths and rotations. One example is shown in Fig. 7. Similarly to the results in Fig. 5, the decomposition with the 7-primitive set produced schematic and abstract interpretations as that in Fig. 7B. A straight primitive was frequently used to approximate a demonstration with a low curvature. Alternatively, a combination of a straight and a curved primitive was used to interpret an unknown curvature as in the lower left part of Fig. 7B. The analysis revealed that, although the demonstrations are exact and without noise, the algorithm could not always find the exact primitives that were used to create the demonstration: similar primitives could occasionally be used to create good but not exact reproductions. This is an expected consequence of the fact that the algorithm does not minimise the error between demonstration and reproduction. The test shows that the algorithm expresses its full potential in reconstructing corrupted data (Fig. 5) rather than reproducing precise demonstrations.

3.2 Decomposition of hand writing

The decomposition of human-generated writing trajectories is a task in which the symbolic aspect is more important than the exact geometry. In other words, global features in a trajectory are fundamental in distinguishing different letters more than the precise geometry of the trajectory. The proposed algorithm was shown in the previous section to be suited to extract high level representations from noisy data. It is natural to ask whether this feature may be of use as a step towards abstracting human hand writing. Note that the experiment in this section decomposes and represents hand writing as a set of primitives, but it does not interpret or map trajectories to letters.

Two examples of human writing data were analysed. A first word "Hello" was decomposed as shown in Fig. 8. The first row (Fig. 8A) is the original trajectory. Fig. 8B is the first approximation, i.e. one single primitive. Fig. 8C shows the representation after 7 steps. The algorithm has identified some of the main features of the demonstration. Fig. 8C illustrates the 9th step, demonstrating how each step is functional in discovering further features. The letter 'h' and one 'l' are already readable after 9 steps. Fig. 8E shows the final reproduction.

The decomposition proved robust with respect to different trajectory matching criteria (Eqs. 1-3) and precision parameters. Fig. 9 shows the final decomposition of the word

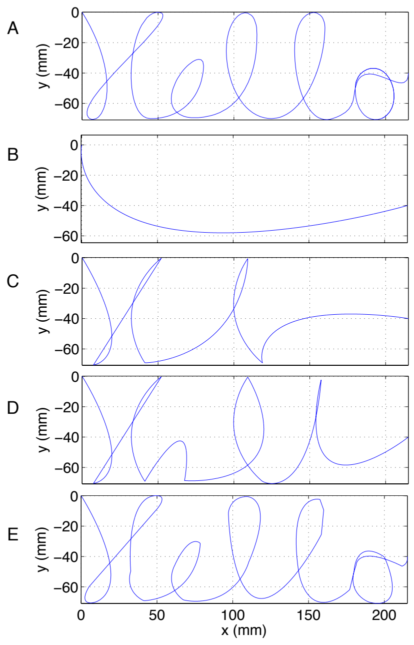

Figure 8 Decomposition of the word "Hello". (A) The demonstrated trajectory. (B) First iteration: the complete trajectory is approximated by one single stroke, i.e. one primitive chosen in the MJM-51 set. (C) Iteration 7: the algorithm has identified main features in the demonstration. (D) Iteration 9: at each further step, more features are captured and represented. (E) The final decomposition and representation.

"Hello" with various criteria. Fig. 9A uses Eq. 3 as matching criteria (i.e. the area between trajectories). Fig. 9B is a decomposition with trajectory matching criterion Eq. 1. And finally, Fig. 9C is a decomposition with Eq. 2 and a lower precision $p = 20$. The reproductions in Fig. 9 are similar but not identical; also compare with Fig. 8E in which Eq. 2 and precision $p = 40$ were used. It can be inferred that different matching criteria and precision parameters affect the decomposition but do not change significantly the capability of the algorithm to represent a demonstration.

Fig. 10 illustrates the decomposition process for the word "Amarsi", whose original hand writing is plotted in Fig. 10A. Fig. 10B and C show iterations two and four during decomposition with the 51-primitive set. Also in this case the algorithm begins by reproducing the most relevant features of the demonstrated trajectory. Note that the way the algorithm proceeds is determined by the tentative segmentation points discovered with the criterion of the maximum error between demonstration and reproduction. The hypothesis is that this criterion, although trivial and of simple implementation, is nevertheless effective in finding progressively prominent features of a demonstration. The illustration of the step-wise iteration demonstrates exactly that. A video showing the complete decomposition process is provided as support material. Fig. 10D shows the final approximation with the large 51-primitive set. Fig. 10E shows the reproduced trajectory as it was decomposed by the 7-primitive set. In this case, the approximation appears less accurate and straight lines are frequently used. How-

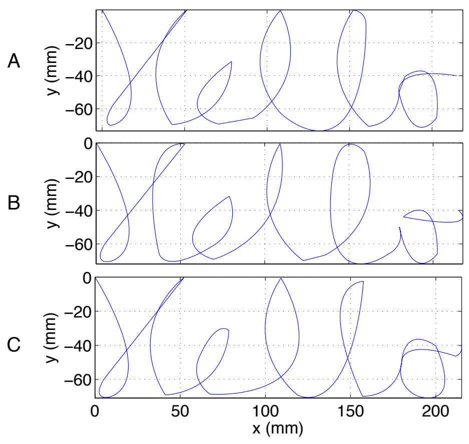

Figure 9 Decomposition of the word "Hello" with different settings. (A) Final decomposition with Eq. 3 as matching criterion (area between trajectories). (B) Final decomposition with Eq. 1 (maximum error). (C) Final decomposition with Eq. 2 and precision $p = 20$. The decompositions are slightly different in each case, however, the capability of decomposing and representing the demonstration appear robust with respect to different matching criteria and precision of primitives.

ever, the main features of the demonstration are captured indicating that the smaller set of primitives resulted in a more abstract representation. It is interesting to note that the decomposition with the larger primitive set results in better approximation with fewer parts. Although this fact appears intuitive, these experiments show that the current method achieves this trade-off that emerges autonomously when the set of primitives changes. It can be concluded that the proposed method adapts autonomously to exploit the specific primitives, i.e. the available skills, of the agent or robotic platform that performs the movements.

The reconstruction capabilities, already proven earlier with the test in Fig. 5, are preserved also when decomposing and reconstructing longer trajectories. A decomposition was run on the data-set in Fig. 10A, corrupted by the addition of $\pm 1\%$ noise to each sampling point. The decomposition proceeds on this noisy data-set similarly to the case without noise. Fig. 11 shows that the reproduced trajectory is an interpretation of the noisy data. This simulation proves that the proposed algorithm employs its generalisation capabilities to filter noise and detect relevant features in the demonstrated trajectory.

4 Discussion

The original idea in the proposed algorithm is to decompose an arbitrarily complex trajectory using the agent's pre-learnt primitives during an iterative process of learning-by-doing. The process starts with a rough approximation of the demonstrated trajectory and learns step by step the features of the input data by a progressive decomposition. Segmentation points are discovered simply by a criterion of maximum error between demonstration and reproduction. Such a trivial criterion that ignores features of both

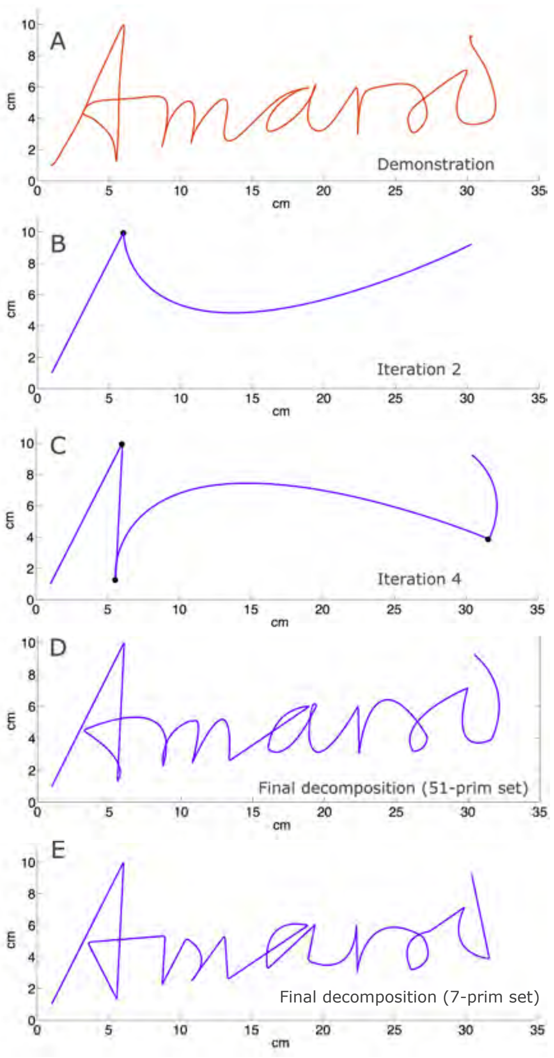

Figure 10 Decomposition of the hand-written trajectory "Amarsi". (A) The demonstrated trajectory recorded from handwriting. (B) Iteration 2 during the composition with the large 51-primitive set in Fig. 1B. (B) Iteration 4 during the decomposition. (C) The final decomposition and approximation using the set of 51 primitives. The demonstration was decomposed in 18 parts (D). The final decomposition and approximation using the set of 7 primitives in Fig. 1A. The algorithm decomposed the trajectory in 21 parts. Although the accuracy with the 7-primitive set is lower than with the larger set, the reproduction appears readable and to some extent a more abstract representation of the demonstration.

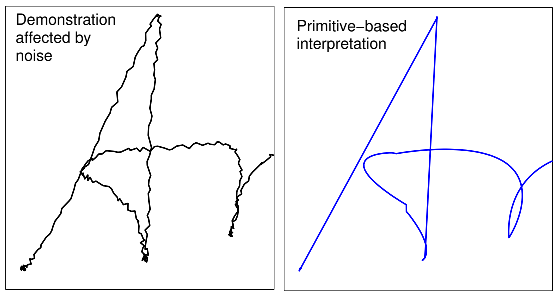

Figure 11 Detail of the decomposition of a noisy version of the hand-written trajectory "Amarsi". (Left) Noisy demonstration. (Right) Reconstruction using the 51-primitive set.

demonstration and reproduction proved nevertheless surprisingly effective and robust. The final result is a sequence of primitives that is in effect an intelligent reading of a demonstrated trajectory represented as a general and abstract concept. The strength of the algorithm lies in the primitive-centred and progressive search, which uses existing skills and implicitly solves data-induced problems like noise and discontinuous derivatives.

Finding segmentation points and fitting sub-trajectories is potentially an intractable problem if considered exhaustively. The proposed method suggests candidate segmentation points taking advantage of progressive approximations. The computation required to generate a reproduction increases with the number of iterations and the number of available primitives. The removal of constraints, i.e. the search of primitives that bypass segmentation points, is done at a the computational cost of matching the locally segmented demonstration with primitives. However, removing segmentation points results in more general solutions, which justify the additional computation. The removal of constraints is effectively a search procedure to avoid local minima in a highly dimensional search landscape.

For simplicity, the current study considers finite sets of primitives in which each primitive has a fixed geometry. An alternative approach consists in using primitives with variable geometry that use one parameter to change certain features as, for example, the curvature. The use of infinite-set primitives requires a different representation, but does not increase the computational complexity of the search. In fact, a larger variety of geometries can be implemented with fewer tuneable primitives. The extension of the algorithm to infinite-set primitives is promising particularly in the cases where high precision and compact representations are required.

The algorithm appears to have generalisation capabilities even if it decomposes trajectories from one single demonstration. The generalisation capability, noticeable particularly in Fig. 5 (rows 1 and 3), derives from the interpretation of the demonstration according to the agent's set of primitives, and less emphasis on the original data. The reconstruction from noisy data in particular shows the generalisation capability in reconstructing straight lines, identify correct curvatures, as well as maintaining cuspids, as clearly shown in Fig. 11.

The criteria upon which the algorithm is constructed

(Sec. 2) represent the *intelligence* of the decomposition, which is intended to mimic loosely human processes of understanding, acquiring and reproducing articulated movement or trajectories. For this reason, the proposed algorithm focuses less on the input data itself and more on the quality of the procedure applied to interpret it. The use of primitives implies inevitably the classification of imprecise and noise-affected demonstration into well defined trajectories. Therefore, such a process causes the loss of accuracy from the demonstrated data. However, such an accuracy may not be descriptive of features of the demonstration. Abstract representations are more compatible with hypotheses on how humans and animals represent and execute movements.

The precision parameter, encoding the second dimension or thickness of a primitive, determines effectively to which level small details in the demonstration need to be reproduced. As a consequence, high precision means that a noisy demonstration is reproduced accurately down to small details, while low precision means that the trajectory is more heavily interpreted according to the agent's primitives. It is important to note that a low precision parameter is not equivalent to high noise filtering. In fact, cuspids and prominent features of the demonstrations are nevertheless captured as shown in Fig. 11.

The method is tested here using one demonstration only for each trajectory. A promising extension is to use multiple demonstrations of the same trajectory to increase the generalisation properties of the algorithm. In particular, more observations of one demonstration are likely to have variations but retain relevant features. One extension is to increase its capability of generalising trajectories by finding one decomposition of a set of similar demonstrations.

The algorithm uses primitives and demonstration in a two dimensional space. The method can be extended and applied to a 3D scenario because primitives and matching functions can be equally generated and computed in 3D space. The increased dimensionality implies also a larger search space, extended sets of primitives and more computation required. It is conceivable that primitives in a 3D space may nevertheless lay on a two-dimensional plane, and that truly 3D trajectories like a helix are relatively rare. The extension presents challenges but is a promising venue for reproducing fully-fledged robotic movements in space.

The trajectories considered in this study were only determined by the geometry without velocity profiles. In effect, releasing the constraints on velocity allows agents to reproduce complex demonstrations by freely choosing from their own primitives with given velocities representing their own capabilities. Extensions of the algorithm could include velocity profiles. The addition of kinematics may imply that velocity cannot drop to zero at segmentation points, introducing strict constraints to the search. In effect, whereas kinematics are essential in dynamics movements such as walking, they become less stringent in object manipulation and marginal in drawing and writing. As the respect of kinematics constraints depends heavily on the precise field of application, tailored algorithms may be required.

The proposed method focuses on the decomposition of trajectories and does not consider the learning of new primitives. The results in this paper showed that the set of primitives is important to achieve particular required performance, and in particular is crucial in interpreting noisy or corrupted data. It is natural to ask how the algorithm can be adapted to extend the available set of primitives while decomposing. A promising research direction is that of integrating the current method in a more powerful algorithm that learns additional primitives with experience. Additionally, certain sequences of primitives that repeat themselves frequently could be assimilated as a new longer primitive, thereby accelerating the search in future occurrences of the given sequence.

The variety of tasks in which simple movements are combined to achieve complex movements extends to numerous scenarios. The proposed method can be applied to those scenarios in which imprecisely perceived movements need to be decomposed, learnt and reproduced. In particular, robots with different dimensions, joint structures and degree of freedom can attempt to perform complex movements according to their own features and capabilities. The implication is that the current method, by adopting an agent-centred and iterative approach to decomposition, is suited to a large variety of robotic platforms, particularly animal-like and humanoid robots that are required to perform a large variety of tasks, not all of them perfectly fitting their anatomical features.

5 Conclusion

A new approach to decompose and reconstruct complex trajectories is proposed. The method starts decomposing a complex trajectory with one initial single primitive and progressively increases the accuracy of the approximation through an iterative process. This approach allows for an initial reduction of the search space with the identification of prominent features of a demonstrated trajectory. Subsequently, the iterative search makes use of newly found segmentation points to search locally better solutions and escape local optima. The agent-centred process offers a new way of interpreting data as function of the agent's skills, which may represent various optimal primitives generated with established methods. The algorithm proves robust and displays remarkable generalisation and feature extraction capabilities. In particular, the algorithm is suited to reconstructing trajectories from corrupted and noisy data. Diverse robotic platforms with different degrees of accuracy and motor patterns could benefit from this method while learning progressively and autonomously the decomposition of complex trajectories. Promising extensions of the algorithm include the applications to a variety of tasks such as imitation learning, learning of complex motor patterns, gestures, object manipulation, software-based and robotic hand-writing.

Acknowledgement

This work was supported by the European Community's Seventh Framework Programme FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics under grant agreement No 248311 - AMARSi. The authors thank William Land for discussions and comments on earlier version of the manuscript.

Appendix

5.1 Generating the primitives with the minimum-jerk model (MJM)

The minimum-jerk model (MJM) [36] is used in the current study to generate two primitives sets (compare Fig. 1A and B) and sample trajectories for the testing in Section 3.1. This model plans a trajectory starting from a given starting point to a given end-point through a via-point. The constraint for planning the trajectory is to be as smooth as possible (minimum jerk). In the generated data-sets the via-point is located at the maximum of each shape, which is reached at $t = 0.5$ of the movement duration.

5.2 Generating trajectories with ELM

A point-to-point motion is driven by a vector field (i.e. a mapping from position $\mathbf{x}$ to vector $\mathbf{v}$) represented by a data driven learning method called Extreme Learning Machine (ELM). The ELM is a feedforward neural network [40] that comprises three different layers of neurons: the input layer $\mathbf{x} \in \mathbb{R}^I$, the hidden layer $\mathbf{h} \in \mathbb{R}^R$, and the output neurons $\mathbf{v} \in \mathbb{R}^I$. The input is connected to the hidden layer through the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R\times I}$ that is unchanged after random initialisation. A supervised learning schema [41] was adopted to compute the output weight matrix to generate stable movements. Each new primitive is learnt from at least three human demonstrated trajectories. The sequence of motion can be computed by discretisation of $\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$, where $\mathbf{x}(0) \in \mathbb{R}^d$ denotes the starting point. Different movements can be generated depending on the starting point relative to the target. The mean starting point $\mathbf{x}_{ms}$ of all demonstration was used to learn each primitive. Using such a starting point, the learn primitive is likely to be similar to the average demonstration. The movement is then normalised such that the start point is at $\mathbf{x}_{ms} \rightarrow \mathbf{x}_S = (-1, 0)$ and the end point at $\mathbf{x}_T = (0, 0)$. The normalised primitives are used in the algorithm with correct rotations and scaling accordingly to the initial (I) and final (F) points as described in the paper.

References

[1] T. Schack, "The cognitive architecture of complex movement," *International Journal of Sport and Exercise Psychology*, vol. 2, no. 4, pp. 403–438, 2004.

[2] J. Vauclair, "Phylogenetic Approach to Object Manipulation in Human and Aple Infants," *Human Development*, vol. 27, pp. 312–328, 1984.

[3] N. Bernstein, *The Coordination and Regulation of Movements.* Oxford: Pergamon Press, 1967.

[4] T. A. Easton, "On the normal use of reflexes: The hypothesis that reflexes form the basic language of the motor program permits simple, flexible specifications of voluntary movements and allows fruitful speculation," *American Scientist*, vol. 60, no. 5, 1972.

[5] T. Flash and B. Hochner, "Motor primitives in vertebrates and invertebrates," *Current Opinion in Neurobiology*, vol. 15, no. 6, pp. 660–666, 2005.

[6] A. d'Avella, P. Saltiel, and E. Bizzi, "Combinaitons of muscile synergies in the construction of a natural motor behavior," *Nature Neuroscience*, vol. 6, pp. 300–306, 2003.

[7] C. B. Hart and S. F. Giszter, "A Neural Basis for Motor Primitives in the Spinal Cord," *The Journal of Neuroscience*, vol. 30, no. 4, pp. 1322–1366, 2010.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. ICRA*, vol. 2, 2002, pp. 1398–1403.

[9] S. Schaal, "Dynamic movement primitives -a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines.* Springer, 2006, pp. 261–280.

[10] S.-M. Khansari-Zadeh and A. Billard, "BM: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models," in *Proc. ICRA*, 2010, pp. 2381–2388.

[11] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *Robotics, IEEE Transactions on*, vol. 27, no. 5, pp. 943 –957, 2011.

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: A new learning scheme of feedforward neural networks," in *IEEE Intern. Joint Conf. on Neural Networks*, 2004, pp. 985–990.

[13] R. F. Reinhart and J. J. Steil, "Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot iCub," in *Proc. Humanoids*, 2009, pp. 323–330.

[14] ——, "Neural learning and dynamical selection of redundant solutions for inverse kinematic control," in *Proc. Humanoids*, 2011, pp. 564–569.

[15] R. Mann, A. D. Jepson, and T. El-Maraghi, "Trajectory segmentation using dynamic programming," in *Proc. International Conference on Pattern Recognition*, vol. 1, 2002, pp. 331 – 334.

[16] J. Kohlmorgen and S. Lemm, "A Dynamic HMM for On-line Segmentation of Sequential Data," in *Proc. NIPS*, vol. 14, 2001, pp. 793–800.

[17] S. Hellbach, J. P. Eggert, E. Koerner, and M.-H. Gross, "Basis Decomposition of Motion Trajectories using Spatio-Temporal NMF," in *Proc. ICANN*, vol. 5769, 2009, pp. 804–814.

[18] V. Mohan, P. Morasso, J. Zenzeri, G. Metta, V. S. Chakravarthy, and G. Sandini, "Teaching a humanoid robot to draw 'Shapes'," *Autonomous Robots*, vol. 31, no. 1, pp. 21–53, 2011.

[19] D. M. Endres, Y. Meirovitch, T. Flash, and M. A. Giese, "Segmenting sign language into motor primitives with bayesian binning," *Frontiers in Computational Neuroscience*, vol. 7, no. 68, 2013.

[20] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen, "Constructing skill trees for reinforcement learning agents from demonstration trajectories," in *Proc. NIPS*, vol. 23, 2010, pp. 1162–1170.

[21] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[22] S. Schaal and C. G. Atkeson, "learning control in robotics – trajectory-based opitimal control techniques," *Robotics and automation magazine*, no. 2, pp. 20–29, 2010.

[23] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Proc. ICRA*, 2011, pp. 3828–3834.

[24] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[25] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila, "Primitives for smoothing mobile robot trajectories," *Robotics and Automation, IEEE Transactions on*, vol. 11, no. 3, pp. 441–448, 1995.

[26] T. Kulvicious, K. Ning, M. Tamosiunaite, and F. Woergoetter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *Robotics, IEEE Transactions on*, vol. 28, pp. 145–157, 2012.

[27] T. Jerde and M. Flanders, "Coarticulation inf fluent fingerspelling," *Journal of Neuroscience*, vol. 23, no. 2383-2393, 2003.

[28] R. Sosnik, B. Hauptmann, A. Karni, and T. Flash, "When practice leads to co-articulation: the evolution of geometrically defined movement primitives," *Experimental Brain Research*, vol. 156, pp. 422–438, 2004.

[29] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2011.

[30] A. Lemme, K. Neumann, F. R. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields," in *Proc. ESANN*. d-facto, 2013, pp. 327–332, best paper award.

[31] C. Atkeson and J. McIntyre, "Robot trajectory learning through practice," in *Proc. ICRA*, vol. 3, 1986, pp. 1737–1742.

[32] Y. Wada and M. Kawato, "A theory for cursive handwriting based on the minimization principle," *Biological Cybernetics*, vol. 73, no. 1, pp. 3–13, 1995.

[33] A. Soltoggio, A. Lemme, and J. Steil, "Using movement primitives in interpreting and decomposing complex trajectories in learning-by-doing," in *Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO 2012)*, 2012, pp. 1427 – 1433.

[34] N. Tsakarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. Ijspeert, M. Carrozza, and D. Caldwell, "iCub - The Design and Realization of an Open Humanoid Platform for Cognitive and Neuroscience Research," *Journal of Advanced Robotics, Special Issue on Robotic platforms for Research in Neuroscience*, vol. 21, no. 10, pp. 1151–1175, 2007.

[35] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Sch?ffer, D. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and K. R. Gmbh, "Lightweight robot arm: a new reference platform for robotics research and manufacturing," in *Joint 41th International Symposium on Robotics and 6th German Conference on Robotics*, 2010, pp. 1–8.

[36] T. Flash and N. Hogan, "The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[37] S. Edelman and T. Flash, "A model of handwriting," *Biologicial Cybernetics*, vol. 57, pp. 25–36, 1987.

[38] S. Edelman, "Representation is representation of similarities," *Brain and Behavioural Sciences*, vol. 21, no. 4, pp. 449–467, 1998.

[39] Y. Wada, Y. Koike, E. Vatikiotis-Bateson, and M. Kawato, "A computational theory for movement pattern recognition based on optimal movement pattern generation," *Biological Cybernetics*, vol. 73, no. 1, pp. 15–25, 1995.

[40] Z. Huang and C. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[41] K. Neumann, A. Lemme, and J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *IROS*, 2013, to appear.

Andrea Soltoggio received his B.Sc. and M.Sc. equivalent degree in Computer Engineering in 2004 from the Norwegian University of Science and Technology, Norway and from Politecnico di Milano, Italy, and his Ph.D. in Computer Science in 2009 from the University of Birmingham, UK. He worked with the Laboratory of Intelligent Systems at EPFL, Lausanne, CH, in 2006 and 2008-2009. He was a visiting researcher at the University of Central Florida, U.S., in 2009 and took the position of technical coordinator of the EU-funded FP7-IP project AMARSi in 2010 with the Research Institute for Cognition and Robotics, Bielefeld University, Germany. His research interests encompass evolutionary computation, learning and plasticity in neural networks as well as broader aspects of cognition and intelligence. E-mail: asoltogg@cor-lab.uni-bielefeld.de (Corresponding author)

Andre Lemme studied computer science at Bielefeld University. He received the diploma in September 2009. In his diploma thesis he worked on learning efficient sparse code in a full non-negative auto associative neural network. In 2009 he joined the Research Institute for Cognition and Robotics (Cor-Lab), where he continues his research on neural networks. Since 2010 he is involved in the EU-funded project AMARSi and was a visiting researcher at the Weizmann Institute of Science, Israel, in 2011. His main interests are algorithms for autonomous learning and developing rich motor skills. E-mail: alemme@cor-lab.uni-bielefeld.de