Proceedings of the RAAD 2008
17th International Workshop on Robotics in Alpe-Adria-Danube Region
September 15-17, 2008, Ancona, Italy

Dynamical system for learning the waveform and frequency of periodic signals – application to drumming

Andrej Gams[a], Sarah Degallier[b] , Auke J. Ijspeert[b] and Jadran Lenarčič[a]

[a] *Department for Automation, Biocybernetics and Robotics, "Jožef Stefan " Institute – Ljubljana,Slovenia – E-mail: andrej.gams@ijs.si URL: http://abr.ijs.si*

[b] *Biologically Inspired Robotics Group, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne – Lausanne, Switzerland URL: http://birg.epfl.ch*

Abstract. The paper presents a two-layered system for learning and encoding a periodic signal and its application to a drumming task. The two layers are the dynamical system responsible for extracting the main frequency of the input signal, based on adaptive frequency oscillators, and the dynamical system responsible for learning of the waveform with a built in learning algorithm. By combining the two dynamical systems we can rapidly teach new trajectories to robots. The systems works online for any periodic signal, requires no signal processing and can be applied in parallel to multiple dimensions. Furthermore, it can adapt to changes in frequency and shape, e.g. to non-stationary signals, and is computationally inexpensive. The algorithm is demonstrated in a drumming task using the HOAP-2 humanoid robot.

Keywords. Dynamical system, Learning, Frequency adaptation, Drumming

1. Introduction

This paper presents a new system for online learning and encoding periodic signals, and its application to the learning and repeating of the trajectory for a drumming task on a HOAP-2 humanoid robot.

Various methods of performing drumming with robots exist. As it is a typical periodical task, the methods range from simple memorizing of the whole trajectory (An, 1988), on-line generation or modification of trajectories depending on the task and the state of the actuated device (Žlajpah, 2006), or for example generation of periodic trajectories using singular values decomposition and vector fields (Okada, 2002).

Specific methods for the learning of drumming, but also applicable to other periodic tasks were described by, for example, (Degallier *et al.*, 2006), (Kotosaka and Schaal, 2000) and. (Ijspeert *et al.*, 2002).

The approach in (Degallier *et al.*, 2006) deals with rhythmic and discrete trajectory generation using nonlinear oscillators and less with motion imitation. The approach by (Kotosaka and Schaal, 2000) explores synchronisation with external signals using neural oscillators, an approach applicable to many device specific tasks; see also (Williamson, 1998).

The system we will present is an extension of the approach by (Ijspeert *et al.*, 2002), which is based on motion imitation and done by the learning of attractor landscapes of point attractors to form control policies (Ijspeert *et al.*, 2002, a, b). The concept was applied to both discrete (Ijspeert *et al.*, 2001) and periodic movements. The essence of this approach is in anchoring Gaussian basis functions in the phase of a dynamical system, and the learning of a weight vector that multiplies the basis functions thus forming arbitrary smooth new attractor landscapes (Ijspeert *et al.*, 2002, a). Nonlinear regression techniques are employed to learn the weight vector.

The limitation of such an approach for periodic movement is that the period has to be known (Ijspeert *et al.*, 2002b) – either specified beforehand, or extracted from the signal using signal processing, e.g. FFT. In case of the frequency changing over time, one can also use one of the feedback quantities of the actuated system to anchor the weight vector, for

example the measured torque (Gams *et al.*, 2007). This, on the other hand, imposes great restrictions to the generality of the approach.

Our extension is a system that surpasses these shortcomings and further expands the approach of anchoring the weight vector in phase space by employing adaptive frequency oscillators for the canonical dynamical system, anchoring the weight vector to their phase.

Adaptive frequency oscillators have the property to on-line adapt to the frequency of any periodic input signal (Righetti *et al*, 2006, Righetti and Ijspeert, 2006). The advantage of using adaptive frequency oscillators is that the whole process of frequency extraction and adaptation is totally embedded into the dynamics of the adaptive frequency oscillator (Righetti *et al*, 2006) and no free parameters, such as a time window, need to be specified, nor do we need to perform any transformations as for example FFT.

The key idea of our system is the splitting of the two dynamical systems – the canonical dynamical system for the extraction of the period, and the output dynamical system for the learning of the waveform. Combining the two dynamical systems provides us with a system that rapidly encodes trajectories of previously undetermined frequencies, works online, and can cope with non-stationary signals as well as the changing of the waveform.

In the next sections we first present the structure of the proposed system (section 2.1), followed by the presentation of the building-blocks of the system – the adaptive frequency oscillators in a feedback structure as the canonical dynamical system (section 2.2), and output dynamical system for the learning of the waveform (section 2.3). In section 3 we present the experimental results for both simulated (3.1) and hand-generated signals (3.2) and the final system structure. Applying the algorithm to a drumming task on a HOAP-2 humanoid robot is presented in section 4. The paper finishes with a short discussion (section 5).

2. Frequency and waveform learning

In this section we first present the structure of the proposed system, followed by the detailed explanation of the two building blocks: canonical dynamical system for the frequency adaptation, and output dynamical system for the learning of the waveform.

2.1. Structure of the system

Fig. 1 shows the structure of the proposed system for the learning of the frequency and the waveform of the input signal.

We input an arbitrary periodic signal into the canonical dynamical system, which consists of an adaptive frequency oscillator feedback structure of several adaptive oscillators. Even though the system can work in parallel for several dimensions, let us assume for the clarity of the presentation that we have a 1-dimensional signal. The canonical dynamical system outputs the information on the frequency ω and the phase ϕ of the oscillator, which are fed to the output dynamical system that is marked by ILWR for Incremental Locally Weighted Regression and basically it learns the weight vector with which we reconstruct the signal. Additionally, being a dynamical system, it ensures a smooth trajectory. Both frequency adaptation and learning of the weights work in parallel. The output of the algorithm can be for example joint coordinates of the robot or position in task space, and the weight vector w_i, which we can use to replay the learned trajectories.

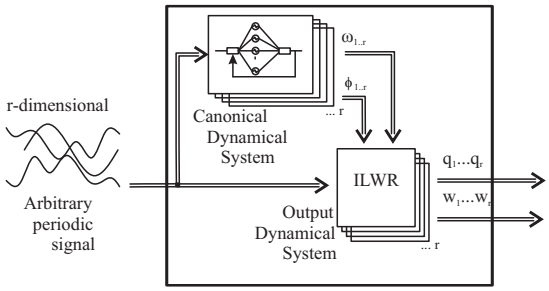

Fig. 1. Proposed structure of the system. The two-layer composition of the system is clearly seen: the Canonical Dynamical System as the first layer and the Output Dynamical System for the learning, as the second layer.

2.2. Canonical Dynamical System

As the basis of our canonical dynamical system we use the phase oscillator (Buchli *et al.*, 2006) to which we apply the adaptive frequency learning rule as introduced in Righetti *et al*, 2006), and combine it with the feedback structure (Righetti and Ijspeert, 2006), which is shown in Fig. 2.

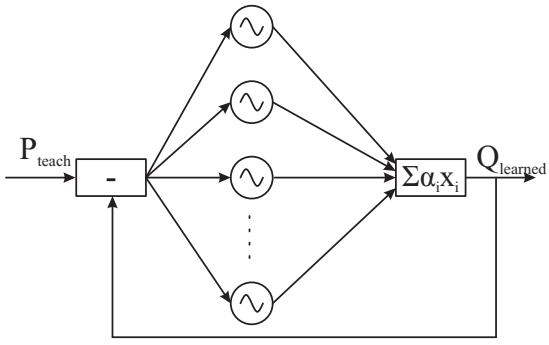

Fig. 2. Feedback structure of a network of adaptive phase oscillators. All oscillator receive the same input and have to be at different starting frequencies to converge to different final frequencies. Refer also to text and eqs. (1-6).

This enables us to adapt to the frequency of the input signal, which is not the case when using a simple nonlinear oscillator to provide a phase signal as in (Ijspeert *et al.*, 2002). The equation of the adaptive phase oscillator in a feedback structure is governed by the following equations:

$$\dot{x}_i = \omega_i - KF(t)\sin(x_i) \quad (1)$$

$$\dot{\omega}_i = -KF(t)\sin(x_i) \quad (2)$$

$$\phi_i = \mathrm{mod}(x_i, 2\pi) \quad (3)$$

$$F(t) = P_{teach}(t) - Q_{learned}(t) \quad (4)$$

$$Q_{learned} = \sum_{i=0}^{M} \alpha_i \cos(x_i) \quad (5)$$

$$\dot{\alpha}_i = \eta \cos(x_i) F(t) \quad (6)$$

where K is the coupling strength or constant, ϕ is the phase, $F(t)$ is the input into the individual adaptive frequency oscillators, $P_{teach}(t)$ is the input signal into the system, $Q_{learned}$ is the weighted sum of the outputs of each oscillator, M is the number of oscillators, α_i is the amplitude associated to the *i*-th oscillator, and η is a learning constant. Throughout the paper we use K=20 and η=1.

Each of the oscillators of the structure receives the same input signal, which is the difference between the signal to be learned and the signal already learned, Eq. (4). As a negative feedback loop is used, this difference approaches zero as the weighted sum of separate frequency components, Eq. (5), approaches the learned signal, and the frequency stabilizes. Such a feedback structure can learn several frequency components of the input signal (Righetti and Ijspeert, 2006) and enables the frequency of the oscillator to converge to the target frequency as $t \to \infty$. Once a frequency of a separate oscillator is set, it disappears from the input due to the negative feedback loop and the other oscillators can adapt to other frequency components. Frequency adaptation results for different signals are presented in Fig. 3.

The top plot shows the adaptation to a non-stationary signal, presented by a chirp signal, to a step change in the frequency of the signal and in the end to a stationary signal. The output frequency stabilizes very quickly at the target frequency. In general this depends on the coupling strength K (Righetti and Ijspeert, 2006).As we can see, the used adaptive frequency phase oscillator can cope with the change in frequency of the input signal.

This proves especially useful when adapting to the frequency of a hand-generated signal, which is never stationary, but always slightly changing. Additionally, we can exploit it to on-line change the frequency of the input signal, allowing us to modify the learning trajectory.

The bottom plot shows the results of the Canonical Dynamical System adapting to an input signal with three frequency components. The

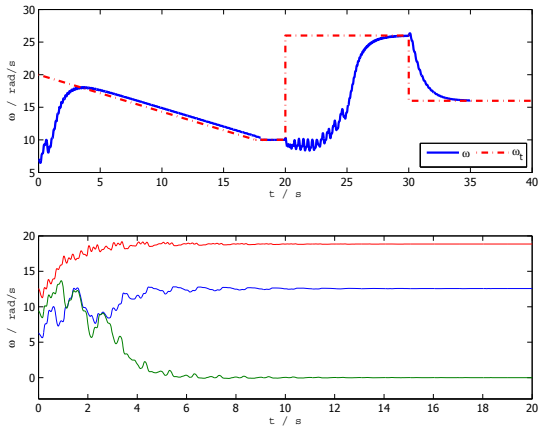

Fig. 3. Top: Adaptation to a chirp signal, ω_0=7rad/s, ω_t=(20-0.6t)rad/s for t<17.6s and ω_t=10rad/s for t>17.6s. Adaptation to step change follows with ω_t = 26rad/s and 16rad/s, step changes occurring at t=20s and 30s. Bottom: Frequency adaptation of a feedback structure with three adaptive phase oscillators to y=0.5+0.8cos(4πt)+2sin(6πt). The initial conditions of the adaptive frequency oscillators are ω0=[3.14, 4.71, 6.28]rad/s. The canonical system successfully adapts to all three frequency components.

oscillators tend to adapt to the components they are closest to, as they fall into their basins of attraction. These are different in size, depending on the power of separate component and the coupling strength, and can even overlap. Fortunately this only poses a design issue, as we have to include enough oscillators in a feedback loop to "eliminate" all the signals.

2.3. Output Dynamical System

The output dynamical system is used to learn the waveform of the input signal. For clarity reasons we present the algorithm for a one dimensional signal.

The following dynamics specify the attractor landscape of a trajectory y towards the goal g with the canonical dynamical system providing the phase signal ϕ to the function Ψ of the control policy

$$\frac{1}{\omega}\dot{z} = \alpha_z(\beta_z(g-y)-z) + \frac{\sum_{i=1}^{N}\Psi_i w_i r}{\sum_{i=1}^{N}\Psi_i} \quad (7)$$

$$\dot{y} = \omega z \quad (8)$$

$$\Psi_i = \exp(h(\cos(\phi - c_i) - 1)). \quad (9)$$

Here g is the goal or the target of the dynamical system and can be used to displace the learned position in space (in our case g=0), ω is the output of the canonical dynamical system, determined by Eq. (2) and N is the number of Gaussian-like kernel functions given by Eq. (9), set to N=25 throughout the paper. α_z=8 and β_z=2, for all the results; the ratio 4:1 ensures critical damping. Such a system without the nonlinear term is a second-order linear system

with a unique globally stable point attractor (Ijspeert *et al.*, 2002).

We use triplets of position, velocity and acceleration ($y_{demo}(t)$, $\dot{y}_{demo}(t)$ and $\ddot{y}_{demo}(t)$) for the input into the learning algorithm. By rewriting Eq. (7) as:

$$\frac{1}{\omega}\dot{z} - \alpha_z(\beta_z(g-y)-z) = \frac{\sum_{i=1}^{N}\Psi_i w_i r}{\sum_{i=1}^{N}\Psi_i}, \quad (10)$$

it can be formulated as a supervised learning problem. By matching y to y_{demo}, z to $\dot{y}_{demo}/\omega$ and $\dot{z}$ to $\ddot{y}_{demo}/\omega$, the target function is then:

$$f_{target} = \frac{1}{\omega^2}\ddot{y}_{demo} - \alpha_z(\beta_z(g-y_{demo}) - \frac{1}{\omega}\dot{y}_{demo}) \quad (11)$$

The learning of the weights is based on the well known locally weighted regression (Schaal and Atkeson, 1998) and is done incrementally for every time sample as governed by equations (12-14):

$$w_i^{t+1} = w_i^t + \Psi_i P_i^{t+1} r(t) e(t) \quad (12)$$

$$P_i^{t+1} = \frac{1}{\lambda}\left(P_i^t - \frac{P_i^{t2} r_i^{t2}}{\frac{\lambda}{\Psi_i} + P_i^t r_i^{t2}}\right) \quad (13)$$

$$e(t) = f_{target}(t) - w_i^t r(t), \quad (14)$$

where Ψi are Gaussian-like kernel functions given in Eq. (9), with c_i equally spaced between 0 and 2π, and h=2.5N determining the width of separate kernel, throughout the paper. w_i are the weights and λ=0.95 is the forgetting factor. The recursion is started with w_i=0 and P_i=1.

Fig. 4 shows the time evolution of the Output Dynamical System with no frequency adaptation, and the weight parameters w_i adjusted to fit the trajectory $y(t) = \sin(2\pi t) + \cos(4\pi t) + \sin(6\pi t - \pi/4)$. As we can see in the top-left plot, the input signal and the reconstructed signal match very closely. The matching between the reconstructed signal and the input signal can be improved by increasing the number of Gaussian-like functions.

3. Experimental evaluation

In this section we present the results of the combined system for both simulated and hand-generated 1D and 2D signals.

3.1. Simulated input signals

Fig. 5 shows the results for a combined system which is learning a signal of shape y=sin(2πt)+sin(4πt+π/3), while at the same time adapting to the frequency. As we can see towards the end of the plots, once the

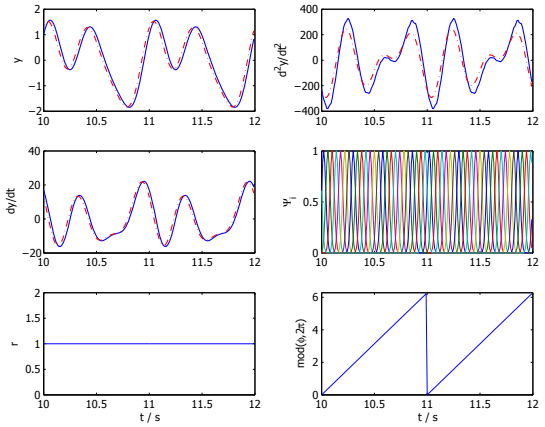

Fig. 4. The result of Output dynamical system with a constant frequency input and with continuous learning of the weights. In all the plots the input signal is the dash-dot line while the repeated signal is the solid line. In the middle-right plot we can see the evolution of the kernel functions and in the bottom right plot the phase of the oscillator. Since we are using the phase oscillator, the amplitude is always r=1, bottom-left.

frequency stabilizes (bottom-left), which is done very fast, so does the phase (bottom-right), and the matching of the input and the learned signal (top-left).

While the frequency is still adapting to the input, as can be seen in the start of the plots, this has a double effect on the generated output. Firstly, the frequency is incorrect and the target trajectory is calculated wrong, and secondly, the phase changes differently in consecutive periods which leads to incorrect

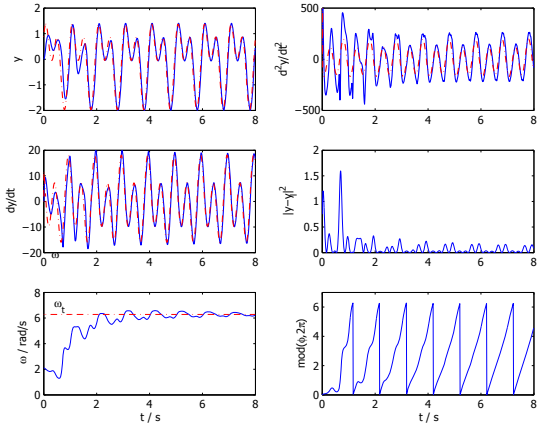

Fig. 5. The result of learning combined with frequency adaptation. Again the input signal is the dash-dot line. The middle-right plot shows the square of the difference of the input and the learned signal. The bottom left plot shows the frequency of the adaptive phase oscillator, for clarity reasons only the common frequency is shown. 2 oscillators were used in the canonical dynamical system; the higher frequency stabilizes at 4π rad/s.

mapping of the input signal to the weight vector of the Gaussian-like functions. With the fast frequency adaptation, the problem disappears by itself.

3.2. Hand-generated signals

We created an interface in Matlab/Simulink that allowed us to input a 2D signal with a computer mouse. We used the algorithm to learn and repeat both dimensions of the signal.

Fig. 6 shows the results for a 2D hand-generated signal with continuous learning of the shape. As we can see in the plots, the learned signal is very close to the input signal. Since we do not input the same signal all the time, but slightly different for every period due to the inaccuracy of the hand, the system continuously adapts to the input signal with a forgetting factor λ, see Eq. (12-14). The setting of the forgetting factor to 0.95 results in a slight delay of the adaptation of the output signal, but still ensures enough "inertia" of the weight vector not to change it completely for every time-step.

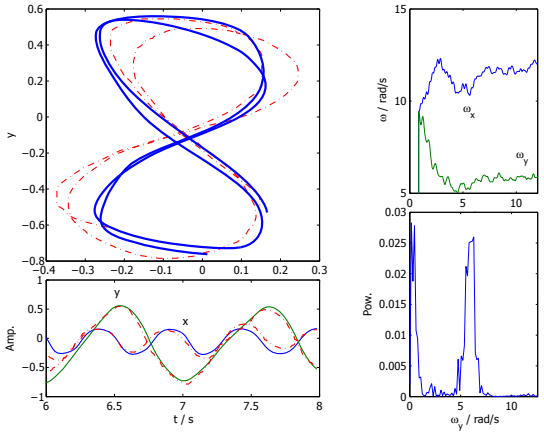

Fig. 6. Results for hand-generated signal with constant learning and frequency adaptation. The top left-plot shows two seconds of the input (dash-dot line) and the learned output signal (solid) in 2D. The bottom-left plot shows 2s of the input signals (solid) and the output signals on a common axis. The top-right plot shows the learned frequencies. The bottom-right plot shows the power spectrum of the *y* dimension of the input signal.

When replaying the signal using a learned set of weights and frequencies, we have to take into consideration that we are trying to recreate two separate signals which belong to the same system. In other words, the frequencies for separate dimensions are in a roughly constant ratio. If this is not the case, the signals drift and the result is not a closed loop. While we do online learning, this is constantly corrected by the learning and the frequency adaptation. When replaying, on the other hand, a common 2D loop has to be considered and the ratio of the frequencies corrected. The correction can be implemented either during the online learning process, or just for the replaying of the signal.

With this, the final control scheme is as presented in Fig. 9. As we can see, we input the r-dimensional signal into parallel r adaptive oscillator feedback structures. The outputs, $\omega_{1\ldots r}$ and $\phi_{1\ldots r}$, are led into a logic block, which determines if the ratio is within the defined interval of an integer number, and outputs a common main frequency $\omega_{m,1\ldots r}$. For the phase, we have two options, as we can either output one common phase for all the r dimensions or we can generate separate phases for separate dimensions. Each has its own advantages. Using a common phase preserves more information, such as slightly different amplitude every other peak, but is for one of the dimensions (the shorter period) stretched so that the periods match. For accurate reconstruction it therefore needs more kernel functions. Using separate phases for separate learning algorithms, on the other hand, requires additional generation of the phases in exactly the previously determined frequency ratio.

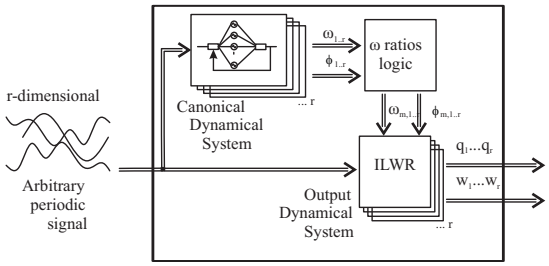

Fig. 7. Modified structure of the system with an introduced logical block to determine frequency ratios.

4. Drumming task

In this section we present the results of applying the algorithm to a drumming task on a humanoid robot HOAP-2.

4.1. Robot control

To control the robot, we separated the interface and the algorithm on one side, and the actual control of the robot's motion on the other side, to two computers as is presented in Fig. 8.

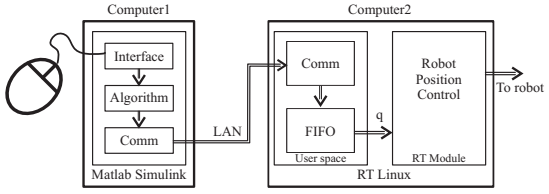

Fig. 8. Schematic of the robot control. We input the signal with a computer mouse in a Matlab/Simulink environment, run it trough the algorithm and send it over the LAN using the UDP protocol to the computer controlling the robot in real time.

On the interface side we record the mouse position and send it to the computer controlling the robot, with a frequency of 100Hz. The robot is position controlled with a frequency of 1000 Hz. Every time the computer controlling the robot receives a new referential position, it copies it to the RT module using a FIFO. As this is 10 times slower than the robot control, the referential positions between two values from computer1 have to be generated by interpolation.

4.2. Inverse kinematics

We mapped the recorded x and y mouse positions to the x (left and right) and y (up and down) positions of the tip of the drumstick attached to the robot's right arm, with a defined value for the third dimension. This requires an inverse kinematics algorithm. As the arm of the robot has 4 DOF we simplified the problem slightly by defining the angle between the horizontal plane and the plane of the forearm-upper arm segments. This allows us to derive an open-loop algorithm for the control of position in task-space.

4.3. Dealing with obstacles

As the drumming requires the contact of the drumsticks and the drum we have to make sure that the referential position is never inside the drum.

To make sure this does not happen we have to modify the Output Dynamical system to include a repulsive force that prevents the robot to pass a predefined limit position. To include a repulsive force we modify Eq. (8) to:

$$\dot{y} = \omega(z + p(y)) \tag{15}$$

where $p(y)$ is defined as

$$p(y) = -\sigma \frac{1}{(y_L - y)^3} \tag{16}$$

with σ=0.0001 determining how close to the limit we can get and y_L being the limit. Fig. 9. shows the results for a 1D signal with the limit set to y_L =[-1,1].

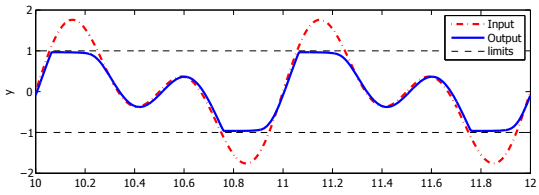

Fig. 9. Output of the system with the limits set to y_L=[-1,1] for the input signal y=cos(2πt)+sin(4πt).

This can be useful for the drumming task in two ways. Firstly, we can set the limit for the physical limit that is the drum, and secondly, we can use this to accurately hit the drum by setting the limit and inputting a signal over the limit. The output signal will only reach the limit and we will be sure we hit the point we want, even though with a slightly different trajectory than we input.

Fig. 10 shows the results for drumming trajectory with the limits for the drums also in the left and right.

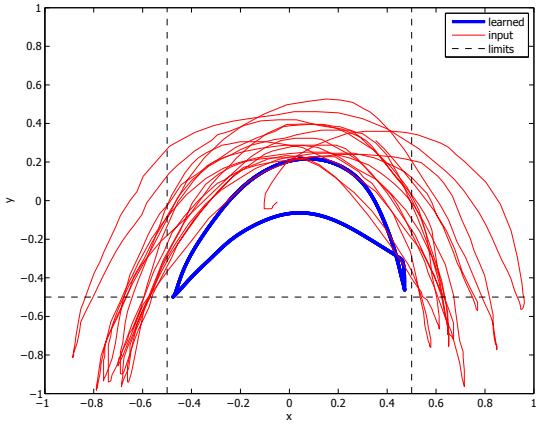

Fig. 10. Results in xy plane for limits set to xl=[-0.5 0.5], yl=[-0.5]. The limits cut the output signal, making this a useful feature for the drumming task.

Fig. 11 shows the achieved robot trajectories for different shapes, which do not need to correspond to drumming as the system can deal with other trajectories as well. Examples of such trajectories are the figure 8 (top right) or a "square". In the bottom-right plot we can see the input and the output signal for the trajectory in the bottom-right.

The setting of the limits needs some calibration of the system, but it could very easily be avoided with some feedback; for the drum a touch sensor would do.

5. Discussion

The described algorithm follows the guidelines for a system for encoding and replaying trajectories. These are: the ease of representing and learning, compactness of the presentation, ease of use for replaying. The system also works as a filter, which makes it suitable for signals with considerable noise, such as signals generated by hand. It works completely online, adapting to stationary and non-stationary signals, and is easy to use. As the output of the system is also the weight vector, we can categorize different trajectories and could in the future use it as means of measuring their similarities.

Another function of the algorithm is also the ability to compress the information of a r-dimensional periodic signal into a vector of size (N x r), where N is the number of Gaussian-like kernel functions. Furthermore, we can reconstruct the signal at an arbitrary frequency by using an oscillator to provide us with the phase.

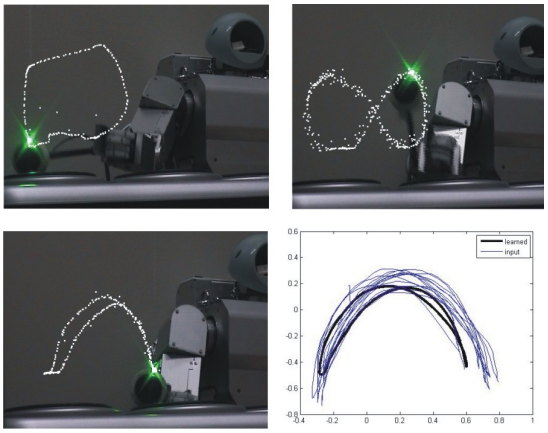

Fig. 11. Repeating different trajectories with the real robot. The system can come also with different trajectories that do not correspond to drumming (top plots) The plot bottom-right is the input and the output trajectories for the plot bottom-left.

We successfully applied the algorithm to the drumming, even if the drumming trajectory is usually not smooth in the point of the hitting the drum and changing direction. Trajectories with "sharp edges" are more difficult to learn, being composed of many frequency components. With the common frequency component learned, the learned trajectory can be very close to the input signal, though maybe a bit smoother around the edges. Instead of using a mouse to input the signal, we can use any other input device with possibility to measure multiple DOF, or even kinesthetic demonstration on the robot itself.

In the future we would like to expand the system to categorize learned trajectories; furthermore, we would like to perform two-handed drumming. A direction to follow is also to try and eliminate the logical block in our structure.

6. Acknowledgments

The work described in this paper was partially conducted within the European Commission's Cognition Unit, project no. IST-2004-004370: RobotCub, and the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

7. References

An, C.H., et al., 1998. Model-based control of a robot manipulator, *MIT Press*.

Buchli, J., L. Righetti, and A.J. Ijspeert, 2006. Engineering entrainment and adaptation in limit cycle systems - from biological inspiration to applications in robotics, *Biological Cybernetics*, Vol. **95**(6), pp. 645-664.

Degallier, S. et al., 2006. Movement generation using dynamical systems: a humanoid robot performing a drumming task, In: *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS06)*.

Gams, A., L. Žlajpah and J. Lenarčič 2007. Imitating human acceleration of a gyroscopic device, *Robotica*, Vol. **25**(4), pp. 501-509.

Ijspeert, A. J. et al., 2001. Nonlinear Dynamical Systems for Imitation with Humanoid Robots, In: *Proc. of the IEEE-RAS International Conference on Humanoid Robots*.

Ijspeert, A. J., J. Nakanishi and S. Schaal, 2002. Learning rhythmic movements by demonstration using nonlinear oscillators, In: *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002)*, pp. 958-963.

Ijspeert, A. J., J. Nakanishi and S. Schaal, 2002a. Learning Attractor Landscapes for Learning Motor Primitives, In: *S. Becker, S. Thrun, and K. Obermayer, editors, Advances in Neural Information Processing Systems 15 (NIPS2002)*, pp. 1547-1554.

Ijspeert, A. J., J. Nakanishi and S. Schaal, 2002b. Movement imitation with nonlinear dynamical systems in humanoid robots, In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA2002)*, pp. 1398-1403.

Kotosaka S., and S. Schaal, 2000. Synchronized robot drumming by neural oscillator, In: *The international Symposium on Adaptive Motion of Animals and Machines*, Montreal, Canada

Okada, M., K. Tatani and Y. Nakamura, 2002. Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion, In: *IEEE Int. Conf. on Robotics and Automation (ICRA 2002)*, pp. 1410-1415.

Righetti, L. and A.J. Ijspeert, 2006. Programmable central pattern generators: an application to biped locomotion control, In: *Proc. of the 2006 IEEE International Conference on Robotics and Automation*.

Righetti, L., J. Buchli and A.J. Ijspeert 2006. Dynamic hebbian learning in adaptive frequency oscillators, *Physica D*, Vol. **216**(2), 269-281, 2006.

Schaal S. and C.G. Atkeson, 1998. Constructive incremental learning from only local information, *Neural Computation*, Vol. **10**(8), pp. 2047-2084.

Williamson, M. M., 1998. Neural control of rhythmic arm movements, *Neural Networks*, Vol. **11**, pp. 1379-1394.

Žlajpah, L., 2006. Robotic yo-yo: modelling and control strategies, *Robotica*, Vol. **24**(2), pp. 211-220.