Imitation Learning for Robot Control: A Review

Hossein Hajimirsadeghi

Abstract—**Imitation plays a crucial role in every-day life of human and animals. Imitation is seen in different levels of abstraction from high level cognitive tasks to low level generation of motor commands. In fact, learning new skills by imitation is faster, safer, and more efficient. Similarly, in robotics and artificial intelligence, due to much amount of conveniences that imitation learning offers to robot programming, it is now a core topic of research in the field. This work presents a survey in imitation learning with more emphasis on robot programming and control. It has been tried to refer to different aspects of the topic even though in a few words, so the readers get familiar with almost all the issues, but the details might be followed in the references. More specifically, learning inverse kinematics, low level learning, and high level learning for the purpose of robot programming by demonstrations are key points which are explained more precisely.**

Index Terms—**Imitation Learning; Robot Programming by Demonstration;**

I. Introduction

IMITAION is one of the main methods of social learning. At the first glance, social learning provides a safer mechanism for learning, but considering complexity of human societies and rapid change of human culture proposes that social learning might be also more efficient [1]. It means that imitation learning can be more efficient in time, energy, etc. than individual learning. There are several types of social learning which are somehow similar to imitation like mimicking or sampling, but according to [2] imitation is discriminated from others by three sources of information: goal, action, and result. In fact, perfect imitation is yielded by understanding the goals of task, regenerating the actions, and making the same results.

Looking into the field of artificial intelligence and robotics, there are the same and even more rationales for imitation learning or its traditional title *programming by demonstration* (PbD). First, imitation makes a powerful tool to speed up learning of complicated tasks in complex robots which might have numerous degrees of freedoms (DOFs) [3]. Indeed, the knowledge transfer in imitation learning reduces the amount of trial and error to make a successful movement [4]. Second, with imitation, artificial agents can produce more humanlike and realistic movements and behaviors. Third, imitation provides an implicit way of training a machine which is a key point in human robot interaction (HRI) [3]. Finally, studying and modeling the coupling of perception-action through imitation might help to understand mechanisms that make developmental organization of perception and action in animals [3].

Robot PbD started in 1980s for the sake of simplicity and cost reduction in manufacturing industries [3]. The early works of Robot PbD was based on symbolic reasoning, e.g. the task was segmented into a sequence of state-action-state, or if-then-rules were employed to describe symbolic relations in graph of states and actions. Then machine learning techniques entered PbD to define a discrete set of basic motor skills [5]. Most progress in recent years has been in devising better interfaces of teaching like vision [6], data gloves [7], laser range finder [8], kinesthetic teaching [9], etc. Also, much amount of interests has been dedicated to biologically inspired approaches for developing more natural computational models of imitation learning in the last decade. Among them conceptual models [10], [11] and neural models [12]-[14] are more popular. In addition to above, the tools to generalize demonstrated movements to new situations have improved and artificial neural networks [15], radial basis functions [16], fuzzy logic [17], hidden Markov models [18], etc. are alternatives to traditional methods. Indeed, imitation learning is not only mimicking or copying demonstrated movements now and tries to address a set of generic questions, namely what to imitate, how to imitate, when to imitate, who to imitate, and how to evaluate a successful imitation [19], [20]. When and who to imitate are questions have been largely unexplored so far [3]. However, for the problems of what to imitate and how to imitate, we should learn the skill and encode it respectively which are explained in many works.

According to [3], currently, there are two main trends of representing a skill: low level learning and high level learning. Low level learning tries to find the mapping between sensory and motor information and performs the encoding at trajectory level. However, high level learning is formed in a more abstract way, and decomposes the skill to a sequence of action-perception units which is referred to as symbolic encoding. Overall structure of representing a skill either at a trajectory level or symbolic level is illustrated in Fig. 1. In addition, advantages and disadvantage of each representation are explained in Table I.

Another problem in Robot PbD which arises according to embodiment and affordance of the agent is correspondence problem [21]:

"Given an observed behavior of the model, which from a

Report was prepared on Sept. 27, 2009.

H. Hajimirsadeghi is a M.Sc. student of control engineering at electrical and computer engineering department, university of Tehran. Student number: 810187579 given starting state leads the model through a sequence (or hierarchy) of sub-goals in states, action and/or effects, one must find and execute a sequence of actions using one's own (possibly dissimilar) embodiment, which from a corresponding starting state, leads through corresponding sub-goals in corresponding states, actions, and/or effects, while possibly responding to corresponding events."

A special case of correspondence problem which is a conventional problem of robotics and is necessarily pervasive in every imitation learning process is inverse kinematics. Inverse kinematics finds appropriate motor information from kinematic sensory information of end-effectors. Although there are classic solutions to this problem, but increasing growth of complex robots in the era of humanoids and androids, necessitate devising techniques and approaches to learn inverse kinematics.

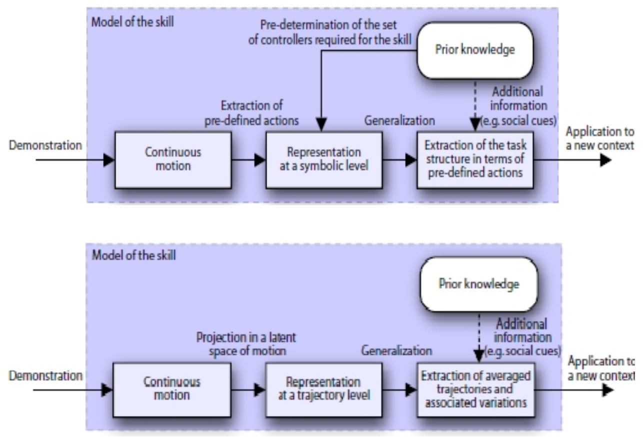
Model of the skill
Pre-determination of the set of controllers required for the skill
Prior knowledge
Additional information (e.g. social cues)
Extraction of pre-defined actions
Generalization
Demonstration
Continuous motion
Representation at a symbolic level
Extraction of the task structure in terms of pre-defined actions
Application to a new context
Model of the skill
Prior knowledge
Additional information (e.g. social cues)
Projection in a latent space of motion
Generalization
Demonstration
Continuous motion
Representation at a trajectory level
Extraction of averaged trajectories and associated variations
Application to a new context

Figure 1. Different levels of representing a skill. Generalization at symbolic level (top). Generalization at trajectory level (down) [3].

TABLE I. ADVANTAGES AND DISADVANTAGES OF REPRESENTING A SKILL AT A SYMBOLIC/TRAJECTORY LEVEL [3]

	Span of the generalization process	Advantages	Disadvantages
Symbolic level	Sequential organization of predefined motion elements	Allows to learn hierarchy, rules and loops	Requires to predefine a set of basic controller for reproduction
Trajectory level	Generalization of movements	Generic representation of motion which allows encoding of very different types of signals/gestures	Does not allow to reproduce complicated high level skills

Structure of this survey is as follows. Section II provides some points about inverse kinematics, comprising importance of learning inverse kinematics, explanations on two cases of no explicit model and redundant DOFs, and introduction to some methods and approaches. In section III, low level learning is described with details and some techniques from three different categories, 1) sensory motor mapping, 2) skill encoding based on statistical modeling, and 3) skill encoding based on dynamical systems are reviewed. Section IV is dedicated to high level learning and contains examples of methods for encoding and representing the skills with different symbolic topologies and frameworks. Finally, conclusions are drawn in section V and some open issues in imitation learning are listed.

II. Learning Inverse Kinematics

Inverse kinematics is a permanent problem in robotics which tries to provide appropriate motor commands in joint space to achieve desired kinematics for the end-effectors of robot in the task space. In classical robotics, redundant actuator systems have mostly been avoided in favor of robots with minimal degree of freedoms (DOFs) and analytical solutions to inverse kinematics [22]. However, with increasing interests to humanoid, assistive, and entertainment robots working in complex environments, there is an emergence of more physically complicated robots with a large number of DOFs [22]. Note that even if there are no extra DOFs, multiple solutions can exist that necessitate more considerations [23]. So we are in need of stronger tools to find inverse kinematics solutions for these robots. In this respect, two main challenges arise. One is to deal with robots which their model cannot be expressed explicitly (or we know nothing about their model) and the other is to come up with singularity of inverse kinematics in highly redundant robots (for which we might know the forward model). For the former, we should take advantage of data driven approaches to approximate the model at least for a limited task space. However, the latter can be solved with more systematic techniques like fixing redundant joints, optimization in local null space of Jacobean matrix, or operational space control methods. A large group of methods categorized in the second approach are local methods suited for online calculations which compute an optimal change in θ (joint angles) and $\Delta\theta$ for a small change in x (end-effectors position) and Δx and then integrate $\Delta\theta$ to make the entire joint space [24].

There are still two useful things about learning inverse kinematics. One is self-calibrating feature of learning, e.g. in the case of calibration errors in camera coordinates, and the other one is that learning works out of experiments, so faces physically correct data and avoids problems of kinematic singularities [23]. In this section we review some recent works which have addressed learning inverse kinematics in the case of challenges.

Lopes et al. proposed a learning approach to find inverse kinematics (called visuo-motor-map (VMM) in the paper) of a humanoid robot in [25]. In this approach, the robot performs arm movement and observes the effect on the image. After gathering data, a multi-layer perceptron (MLP) is used to approximate functions from visual space to motor space. They also comment that it is better to search and tune the most interesting visuo-motor features so that a more compact representation is achieved.

Ajalooeia et al. [26] strengthened the abovementioned idea with biological backgrounds and proposed a more established algorithm for hand-eye coordination in a marionette robot. The algorithm is inspired from babbling in infants where they try to learn the sensory motor systems of their body by performing random movements and following those with interesting effects. The proposed algorithm for hand-eye coordination is as follows. First, a number of temporary goals are determined on the visual path of the teaching trajectory (cf. Fig. 2). Robot starts with an initial joint configuration and makes small perturbations in its joint angles. If this perturbation takes its end-effectors closer to the temporary goal, the movement is performed; otherwise it goes back to the previous state. In this way, the end-effectors sweep all the temporary goals gradually. The visual and motor information at temporary goals are piled up and a mapping form sensory space to joint space is learned with a MLP. Hence, inverse kinematics is locally approximated for the task space.

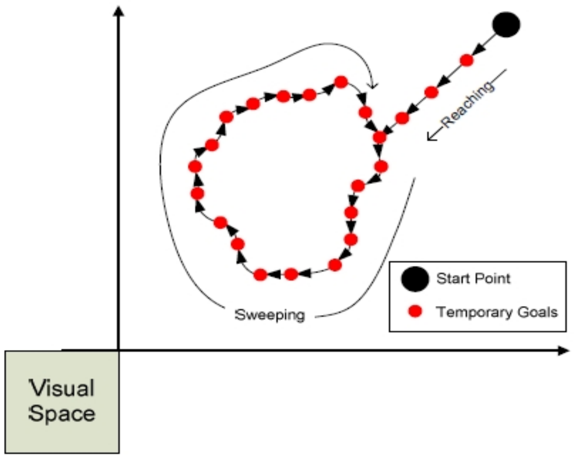

Figure 2. Hand-eye coordination process using babbling [26].

Alissandrakis et al. [27], [28] in their imitation algorithm, ALICE, solved the correspondence problem between dissimilar robotic arms with another mechanism of finding inverse kinematics. In this mechanism, there is still no need of knowledge about model of the system. In their work, actions are discrete values defined as difference between two consecutive joint angles. For example, in [27], the possible actions are 10°, 0° or -10°, so the absolute angle at each joint can be anywhere in the range of 0° to 360°, but only in multiples of 10°. The algorithm is an iterative method composed of choosing between history (correspondence library) and random generation. At each trial (regeneration of a movement), a random action is generated (a different one for each trial). If there is a perceptual match in the corresponding library for current state of the sensory system, another action is also considered. The two actions are then evaluated and compared to each other according to a metric (e.g. similarity in resulting effects or states). Next, the imitator agent performs the better action and corresponding library is also updated. After a number of trials, the imitation is completed.

Lopes et al. [29], in their next work proposed more systematic methods which can also come up with degree of redundancies in robots to learn inverse kinematics within static and dynamic maps. In [29], the forward model is given by

$$I = f(n, r), \tag{1}$$

where r and n denotes motor commands of redundant and non-redundant degrees of freedom respectively, while I is the vector of sensory features. To find inverse kinematics solutions, we are interested in the inverse function, but it is not tractable in a straightforward manner. A common approach is to adhere additional constraints to remove redundancies. For example, the following optimal control problem can be used.

$$(n^*, r^*) = \arg\min_{n,r} \lambda\|I - I^*\| + c(n, r), \tag{2}$$

where I^* is the desired position of end-effector and the regularization term $c(n,r)$ is the control cost that can be "Comfort" (e.g. distance to joint limits), "Energy minimization" (e.g., the position with lower momentum), "Minimum motion" (i.e., minimum motion from current to desired position).

Lopes et al. proposed another approach with the same semantics but different formulation that decreases computational cost significantly. First, the "minimal order sensory-motor map" is defined as $n = g(I, r)$. So computation of nonredundant degrees of freedom becomes well-posed, and only we impose optimization of redundant ones with any control cost function, e.g. maximizing the distance to joint limits (by standing close to central positions)

$$\mathcal{L}(r) = \|g(I^*, r) - n_c\|^2 + \|r - r_c\|^2 \tag{3}$$

where r_c and n_c are central configurations of the corresponding joints. With this cost function r can be find with gradient descent algorithm as follows.

$$r_{t+1} = r_t - \alpha\nabla_r\mathcal{L}(r) \tag{4}$$

If we know nothing about model of robot, the mapping $n = g(I, r)$ can be learned by gathering data from body movements and using a regression algorithm like local weighted regression (LLR), but the workspace should be covered properly in the training set.

Lopes et al. also proposed an algorithm to find dynamic mapping between sensory and motor information. This mapping is more advantageous because it considers the trajectory followed to reach the desired position and has visual feedback. In this approach Jacobean matrix is used for local approximation of the visuo-motor maps, but an idea is also employed to handle redundancy. The idea is to decompose the task into several subtasks which are not disturbing each other. For example, we can have the main task and subtasks like obstacle avoidance or posture definition. Hence, the redundancy is exploited by having several tasks simultaneously performed.

Redundancy formalism for two tasks is as follows. First, there are two tasks e_1 and e_2 and with Jacobean matrices

$$\dot{\mathrm{e}}_\mathrm{i} = \mathrm{J}_\mathrm{i}\dot{\theta} \tag{5}$$

So the control law for the first task will be

$$\dot{\theta} = \mathrm{J}_1^+\dot{\mathrm{e}}_1 + \mathrm{P}_1\mathrm{z}, \tag{6}$$

where P_1 is the orthogonal projection operator on the null space of J_1 and J_1^+ is the pseudo-inverse of J_1. Vector z can be used to accomplish the second task at the same time. By using (6) in (5) we obtain

$$\dot{\mathrm{e}}_2 = \mathrm{J}_2\mathrm{J}_1^+\dot{\mathrm{e}}_1 + \mathrm{J}_2\mathrm{P}_1\mathrm{z} \tag{7}$$

For which the inversed will be

$$\dot{\theta} = \mathrm{J}_1^{+}\dot{\mathrm{e}}_1 + \mathrm{P}_1(\mathrm{J}_2\mathrm{P}_1)^{+}(\dot{\mathrm{e}}_2 - \mathrm{J}_2\mathrm{J}_1^{+}\dot{\mathrm{e}}_1) \qquad (8)$$

As P_1 is Hermitian and idempotent, (8) is simplified as

$$\dot{\theta} = \mathrm{J}_1^{+}\dot{\mathrm{e}}_1 + \widetilde{\mathrm{J}_2}^{+}\widetilde{\dot{\mathrm{e}}_2} \qquad (9)$$

where $\widetilde{\mathrm{J}_2} = \mathrm{J}_2\mathrm{P}_1$ and $\widetilde{\dot{\mathrm{e}}_2} = \dot{\mathrm{e}}_2 - \mathrm{J}_2\mathrm{J}_1^{+}\dot{\mathrm{e}}_1$.

Finally, it is noted that Jacobean matrix can be learned to follow our no model assumption. A practically useful method for online estimation of Jacobean based on Broyden update rule [30] is explained in [31], [32]. The Jacobean is estimated iteratively

$$\hat{\mathrm{J}}(t+1) = \hat{\mathrm{J}}(t) + \alpha\frac{(\Delta y - \hat{\mathrm{J}}(t)\Delta\theta)\Delta\theta^{T}}{\Delta\theta^{T}\Delta\theta} \qquad (10)$$

where $\alpha \in [0,1]$ is the learning rate. To move the sytem to the desired end-effector position y^{*}, the following control law is applied

$$\Delta\theta = h\big(\mathrm{J}^{+}(y^{*} - y)\big) \qquad (11)$$

where $h(.)$ can be chosen to ensure an exponential, linear, or any other type of convergence.

Schaal and his colleagues use a kind of inverse kinematics controller (which is an operational space controller) in their recent works about imitation [33], [34]. Nikanishi et al. [22] examined suitability of operational space control methods thoroughly for complex high DOF robots. Operational (or task space) control is about how to resolve redundancy and how to generate appropriate motor commands in configuration space to make end-effector and DOFs achieve a prescribed level of impedance [22]. The general approach is the use of some forms of Jacobean pseudo-inverse with local null space optimization. Three main types of control which are practically or theoretically relevant to robotics are discussed in the paper: 1) velocity-based control; 2) acceleration-based control; 3) force-based control. In this respect, the forward model and differential relations between joint coordinates and operational space coordinates are describe by

$$\mathrm{x} = \mathrm{f}(\theta) \qquad (12)$$
$$\dot{x} = \mathrm{J}(\theta)\dot{\theta} \qquad (13)$$
$$\ddot{x} = \mathrm{J}(\theta)\ddot{\theta} + \dot{\mathrm{J}}(\theta)\dot{\theta} \qquad (14)$$

where θ denotes the joint angle vector and x is the operational space vector. Redundancy resolution at the velocity level is to inverse (13) to compute the desired joint velocities

$$\dot{\theta}_{\mathrm{d}} = \mathrm{J}^{+}\dot{\mathrm{x}}_{\mathrm{d}} + (\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\xi_1 \qquad (15)$$

where J^{+} is the pseudo inverse of J and ξ_1 is an arbitrary vector which is projected to null space of Jacobean matrix by $(\mathrm{I} - \mathrm{J}^{+}\mathrm{J})$ multiplier. Subscript d denotes desired values for the variables. On the other hand, redundancy resolution at the acceleration level is attained by solving (14) to compute joint accelerations from desired task space accelerations:

$$\ddot{\theta} = \mathrm{J}^{+}\big(\ddot{x} - \dot{\mathrm{J}}\dot{\theta}\big) + (\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\xi_2) \qquad (16)$$

where ξ_2 like ξ_1 is an arbitrary vector which control desired acceleration behavior in the null space. For redundancy resolution, an optimization criterion is considered with cost function

$$\mathrm{g}(\theta) = \frac{1}{2}(\theta - \theta_{\mathrm{rest}})^{\mathrm{T}}\mathrm{K}_{\mathrm{w}}(\theta - \theta_{\mathrm{rest}}) \qquad (17)$$

where K_w is positive definite diagonal weighting matrix and q_{rest} is the rest (preferred) posture. This criterion has been shown to be useful for creating human-like movements in anthropomorphic robots [35], but other control cost functions named in the last work by Lopes et al. can be employed too. Details for deriving different controllers formulation from differential relations in (12)-(14), and optimization criterion in (17), are provided in [22]. Here, we just express the simplest and most prevalent formalism known as velocity-based control:

$$\begin{aligned}\dot{x}_r &= \mathrm{J}^{+}\dot{x}_d + K_p(x_d - x),\\ \dot{\theta}_d &= \mathrm{J}^{+}\dot{x}_r - \alpha(\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\nabla\mathrm{g},\end{aligned} \qquad (18)$$

where K_p is a PD gain matrix. In fact, (18) is a composition of the redundancy resolution inverse model in (15) and steepest gradient method in optimization literature. As explained previously, Jacobian matrix can be learned online in these methods.

Properties of some of the methods described in this section are summarized in Table II.

III. Low Level Learning

Low level representation of a skill tries to find a mapping between sensory and motor information, starting from simple reproduction of the motions by gathering outputs of the inverse kinematics problem or coordinating sensory and motor systems to more complicated approaches like skill encoding based on statistical modeling or dynamical systems. For example, Lopes et al. [25] train a robot to learn sensory motor coordination statically or dynamically, and then use this mapping in the developmental imitation process of the robot. On the other hand, statistical skill encoding at trajectory level (joint space or task space) is commonly used to address variability in multiple demonstrations. Determining the range of acceptable trajectories [36], handling uncertainty with spline smoothing techniques [37], computing mean and variance of demonstrated motions [38], encoding temporal and spatial variation of signals with hidden Markov models (HMMs) or Gaussian mixture models (GMMs) and then retrieving generalized trajectories with multiple HMMs [39] or Gaussian Mixture Regression (GMR) [9] are some examples of the statistical encoding techniques used for low level imitation. Aside from these methods which focus on variability of demonstrations, some statistical techniques like singular value decomposition (SVD) can be used to encode and learn single motion demonstration of the teacher [40].

TABLE II. PROPOERTIES OF SOME METHODS OF LEARNING INVERSE KINEMATICS

Proposed by	Method	Advantages	Disadvantages
Lopes et al. 2005 [25]	Data gathered by movements are mapped from visual space to motor space	• Need no knowledge of model	• Not well-established • Needs a lot of explorations • No sensory feedback • Being static (i.e., point to point mapping and not considering trajectories)
Ajalooeian et al. [26]	Inspired from babbling, temporary goals are marked in the trajectory and robot tries to reach the goal by semi-random movements, then mapping from sensory to motors space is performed	• Need no knowledge of model • Well-established with details	• Needs some explorations between temporary goals • No sensory feedback • Being static
Lopes et al. 2007 (static) [29]	Joint angles of redundant DOFs are obtained by optimizing a cost function and then fed into a function that gives nonredundant joint angles from redundant ones and sensory input	• Need no knowledge of model (if $n = g(I, r)$ is learned) • More systematic and less explorative	• The function $n = g(I, r)$ which gives nonredundant joint angles from redundant ones and sensory input should be learned or predetermined • No sensory feedback • Being static
Lopes et al. 2007 (dynamic) [29]	Decomposing task into several subtasks and remove redundancies in pseudo inverse formulations by performing the subtasks simultaneously	• Need no knowledge of model • Having sensory feedback • Being dynamic • Systematic	• Needs prior knowledge how to decompose the task • Jacobean matrices should be learned
Nikanishi et al.[22]	Jacobean pseudo-inverse control with null space optimization in velocity, acceleration or force space	• Having sensory feedback • Being dynamic • Systematic (no exploration)	• Jacobean matrix should be predetermined

Finally, the most promising approach for skill encoding is to embed dynamics of the motions in dynamical systems. Discrete and Rhythmic movements can be generated by shaping attractors of dynamical systems weather in joint space or task space. It is noteworthy that discrete movements are encoded by point attractors while rhythmic movements can be represented by the limit cycle attractors or *adaptive* point attractors. The former approach is realized with central pattern generators (CPGs). CPGs are natural circuits found in animals that can produce rhythmic patterns of neural activity without rhythmic inputs. The advantages and disadvantages of CPGs described in [41] for robot control are listed in Table III.

TABLE III. ADVANTAGES AND DISADVANTAGES OF CPGS [41]

Advantages
Robostness against perturbations
Well-suited for distributed implementation
Having few control parameters
Smoothness because of dynamical system that damps the changes
Well-suited to integrate sensory feedback
Offering good substrate for learning and optimization algorithms
Disadvantages
Methodology to design CPGs for a particular problem is missing
Theoretical foundation describing CPGs is missing

In the following parts, we explain the abovementioned approaches in more details by scrutinizing the most recent and efficient methods. Specifically, we go through the works of Lopes et al. [29] for sensory-motor coordination, Billard and her colleagues [9], [42] for skill encoding at trajectory levels based on statistical modeling and/or dynamical systems, Schaal and his colleagues for motion learning with dynamical systems [33], [34], Ijspeert and his colleagues for dynamical systems encoding and CPG programming [43], [44], and Ajalooeial et al. for CPG modeling [26]. Some properties of the three discussed approaches, sensory motor coordination, statistical encoding, and dynamical encoding are summarized in Table IV.

TABLE IV. COMPARISON BETWEEN THREE DIFFERENT APPROACHES FOR LOW LEVEL REPRESENTAION

Approach	Advantages	Disadvantages
Sensory Motor Mapping	• Simple and general	• Exhaustive explorative learning • Not robust • No ability of generalization • No modulation of parameters
Statistical Encoding	• Generalization from multiple demonstrations	• Not robust • No modulation of parameters
Dynamical Encoding	• Robust to perturbations • Online modulation of parameters and features • Simplicity of integrating sensory feedback	• Theoretical foundations are not complete • No capability of generalization from multiple demonstrations by its own

A. *Sensory Motor Mapping for imitation learning*

In [29], a developmental roadmap is proposed for imitation learning in robots. The stages of development are as follows: 1) Learning about the self; 2) Learning about objects and worlds; 3) Learning about others and imitation. It means that at the first stage, the robot tries to learn different sensory-motor maps (SMMs) in its body. For example, by self exploratory random arm movements, it struggles to make head-arm coordination by creating head-arm maps (the methods to learn static and dynamic maps are described in *Inverse Kinematics* section). Then in the next stage, the robot learns to recognize and grasp objects and also understand object affordances. For instance in object grasping, the robot gazes at the object by moving its head and then use head-arm mapping (which is an SMM) to find the corresponding joint configuration for the grasping task. Finally, in the imitation stage, it observes the demonstrations, transforms the images from allo-coordinates to ego-coordinates by view-point transformation (VPT), and uses the appropriate SMM to perform the imitation. In this way, the robot can minimize the imitation metric described in (19)

$$im = \int\left(VPT(I_a) - I_a^{self}\right) \quad (19)$$

Where I_a shows the image of the demonstration, and I_a^{self} denotes the image of imitator's end-effectors in its ego-coordinates. With appropriate choice of SMM and considering $VPT(I_a)$ as the input, motor actions are obtained as the output of the mapping. Figure 3 shows a brief description of the proposed imitation procedure.

The advantage of this work is simplicity of implementation and generality of the method for applying in different tasks from low level to goal-directed or high-level tasks. On the other hand, one of the most significant disadvantages is the exhaustive exploratory learning process in the first stage (learn about itself) which may take a long time for the robot to pass it. In addition, this algorithm is incapable of trajectory generalization and also is not robust to perturbations.

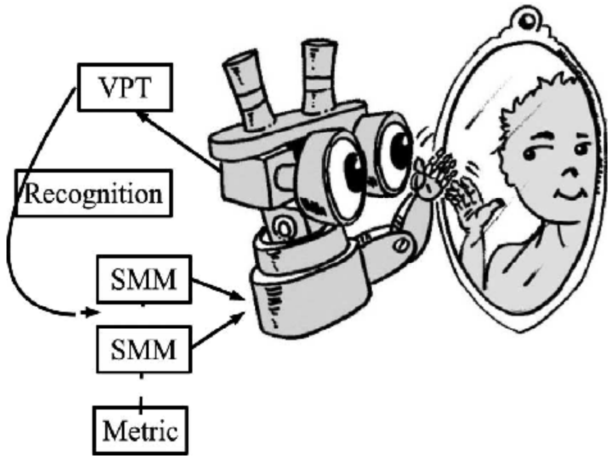

Figure 3. Imitation architecture from [29]. First, observations are transformed to an ego-coordinates (VPT), where segmentation and recognition are made. Then, an imitation metric and body correspondence are chosen (by selecting the corresponding SMM). Finally, the imitation is accomplished.

B. *Skill encoding based on Statistical Modeling*

In this part, we explain the works by Billard and her colleagues [9], and Chella et al [40]. for statistical skill encoding at trajectory level. In [9], Calinon et al. used GMM for encoding trajectories from multiple demonstrations and GMR to make a generalized smooth trajectory for imitation learning. Their work is the basis of the most recent research projects of Billard group for imitation learning which will be discussed in the following parts. Briefly, the algorithm projects the specified trajectories into a latent space of reduced dimensionality, using principal component analysis (PCA). Next, GMM is used to model variability of the signals, and finally, GMR gives a reference trajectory for which an optimal control problem with body (e.g. Jacobian matrix) and environmental constraints (position of the objects) is defined. By solving this problem, an optimal trajectory is obtained which matches the reference trajectory as closely as possible and satisfies the body and environmental constraints at the same time.

More specifically, in [9], a dataset $X_s = \{\theta_s, x_s, y_s, h_s\}$ from N observations of $\theta_s \in R^9$ for joint angles of robot arms, $x_s \in R^6$ for 3D Cartesian position of the two hands, $y_s \in R^6$ for 3D relational position of the two hands from the object, and $h_s \in R^2$ for activity representation of the two hands (grabbing or not) is defined. The data is extracted from multiple demonstrations of a grabbing and moving task. As there are redundant variables in the data, PCA is used to find the most important features and also decrease the dimensionality. The data in the latent space is represented by $\xi_s = \{\xi_s^\theta, \xi_s^x, \xi_s^y, \xi_s^h\}$, and finally, the temporally aligned data $\xi_j = \{\xi_s^j, \xi_t^j\}$ which can be derived by a preprocessing like dynamic time warping (DTW) is extracted where $\xi_t^j \in R$ is the temporal value corresponding to occurrence of the sensory vector ξ_s^j. Note that in the cases where representation of the sensory data with joint angles or Cartesian positions is adequate, there is no need of PCA for redundancy elimination or DTW.

When the temporally aligned data is ready, GMM and GMR are used to encode and generalize the trajectories according to Fig. 4. If we have a binary data like h_s, Bernoulli mixture model can be used instead of GMM. The resulting generalized trajectory is the solution to problem of *what to imitate* which will be contributed to the metric of imitation.

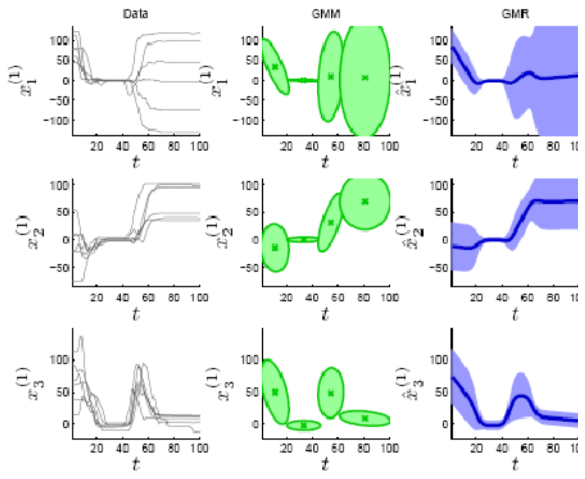

Figure 4. Process of encoding and generalization with GMM and GMR [52].

As it was mentioned before, there are two constraints for this problem: body constraints and environmental constraints. Body constraints are results of kinematic relation between joint angles and end-effectors (Cartesian positions). For the sake of simplicity this relation is characterized with Jacobian matrix in [9]. Environmental constraints are due to relations between x and y variables introduced above ($y_s = x_s - o_s$, where o_s is the initial position of the object). Since the problem and algorithm are defined in the latent space, both two sets of constrains should be rewritten in the latent space. The relations and details are provided in [9]. Note that if the data for trajectories is solely defined in the joint space or task space, there may be no constraints for the problem.

At last the metric of imitation is given by (20) which makes the complete optimal control problem with the body and environmental constraints.

$$\begin{aligned} H &= \left(\xi_s^\theta - \hat{\xi}_s^\theta\right)^T W^T \left(\xi_s^\theta - \hat{\xi}_s^\theta\right) \\ &+ \left(\xi_s^x - \hat{\xi}_s^x\right)^T W^T \left(\xi_s^x - \hat{\xi}_s^x\right) \\ &+ \left(\xi_s^y - \hat{\xi}_s^y\right)^T W^T \left(\xi_s^y - \hat{\xi}_s^y\right) \end{aligned} \quad (20)$$

where $\{\xi_s^\theta, \xi_s^x, \xi_s^y\}$ denotes the candidate trajectories for reproducing the motion, and $\{\hat{\xi}_s^\theta, \hat{\xi}_s^x, \hat{\xi}_s^y\}$ represents the generalized joint angle trajectories. Also, the weighting matrices can be chosen according to covariance matricies of different components as mentioned in [9]. The solution to this problem is presented with details in [9].

In [40], Chella et al. use PCA for encoding and representation of trajectories of different types of motions. As a result, each trajectory is described by a linear combination of basis functions ($\varphi_i(t)$) like (21).

$$\varphi(t) = \sum_{i=1}^{k} c_i \, \varphi_i(t) \quad (21)$$

Where c_i are projection coefficients or singular values in the singular value decomposition process (SVD) used for PCA and k is the number of principal components.

Interestingly, this encoding can be used to make novel trajectories which satisfy new initial and/or terminal conditions. For example, if the trajectory is constructed by three variables $\varphi(t) = [x(t), y(t), \theta(t)]^T$ and k is set to 6 (i.e., six principal components are kept), the following set of linear equations can be solved to attain the new projection coefficients (unknowns).

$$\begin{aligned} \varphi(t_0) &= \sum_{i=1}^{6} c_i \, \varphi_i(t_0) \\ \varphi(t_f) &= \sum_{i=1}^{6} c_i \, \varphi_i(t_f) \end{aligned} \quad (22)$$

Furthermore, coefficients can be employed for classification or abstraction of motion signals.

C. Skill Encoding based on Dynamical Systems

Dynamical systems offer a good substrate for imitation learning by increasing robustness to perturbations and adding capabilities of integrating sensory feedback in the system and online modification of features and parameters.

One of the general ideas for imitation learning with dynamical systems is to use a canonical simple dynamical system with a point attractor and coupling a nonlinear term with unknown parameters to shape the attractor landscapes according to training trajectories. The canonical system is usually the simple second-order spring and damper system and the nonlinear term is a weighted combination of Gaussian basis functions where the weights should be learned [33], [34], [45]. This approach is a commonplace in generation of discrete movements, but periodic movements can also be learned with contribution of phase of oscillation in the nonlinear learning term [43]. It is like that the attractor point of the dynamical system is adapted with the phase of oscillation, so the rhythmic waveform is produced. Another approach for learning of periodic movements is to shape the limit cycle attractor of an inherent oscillator according to training waveforms (CPG programming).

Schaal and his colleagues offered the first works of imitation learning with dynamical systems, and Gams et al. developed it for generation of periodic movements with adaptive frequency oscillators to encapsulate phase of oscillation in the dynamical system. Billard and her colleagues integrated the generalized signals obtained by statistical encoding (cf. part B) with goal-directed point attractor dynamical systems to make more general and robust trajectories. Righetti et al. introduced adaptive frequency oscillators and constructed a generic CPG with coupled adaptive oscillators to reproduce periodic signals. Ajalooeian et al. proposed another CPG programming technique by transforming limit cycle of a base oscillator to the desired trajectory via a multi layered perceptron (MLP). The abovementioned works will be explained with more details in the following paragraphs.

Schall and his colleagues, in their first work, used the nonlinear dynamical system described in (23) [45]. Indeed, this is a simple spring-and-damper system modulated by a nonlinear function which depends on internal states (v, x).

$$\begin{aligned} \dot{z} &= \alpha_z(\beta_z(g-y) - z) \\ \dot{y} &= z + \frac{\sum_{i=1}^{N} \psi_i w_i}{\sum_{i=1}^{N} \psi_i} v \end{aligned} \quad (23)$$

where ψ_i is a Gaussian kernel function given by (24)

$$\psi_i = exp\left(-\frac{1}{2\sigma_i^2}\left(\frac{x-x_0}{g-x_0} - c_i\right)\right) \quad (24)$$

The internal states also have a simple second-order liner dynamics (spring-and-damper system).

$$\begin{aligned} \dot{v} &= \alpha_v(\beta_v(g-x) - v) \\ \dot{x} &= v \end{aligned} \quad (25)$$

This dynamical systems has a unique point attractor at $(z, y, v, x) = (0, g, 0, g)$. Actually, the linear system (second system) is to make v stabilized at zero so that the main nonlinear dynamical system will transform to a linear system (as the nonlinear term becomes zero) with equilibrium at $(z, y) = (0, g)$. Hence, the second system can be replaced by any system that makes v zero at last. In the learning stage, y and z are replaced by demonstrated trajectories and their velocities (in task space or joint space), respectively, and a supervised learning algorithm is used to identify the parameters (e.g. weights) for the nonlinear term. Locally weighted regression (LWR) was the first algorithm employed for learning the parameters and receptive field weighted regression (RFWR) and locally weighted projection regression (LWPR) were the algorithms came in advance for better performance.

Schaal and his colleagues improved the dynamical systems presented in (23) in their next publications and in the most recent ones [33], [34], proposed a bio-inspired dynamical system and facilitated it with an obstacle avoidance coupling. This dynamical system is described in (26):

$$\begin{aligned} \tau\dot{v} &= K(g-x) - Dv - K(g-x_0)\, s + K\, f(s) \\ \tau\dot{x} &= v \end{aligned} \quad (26)$$

where f is defined as

$$\mathrm{f(s)} = \frac{\sum_{i=1}^{N} w_i \psi_i(s) s}{\sum_{i=1}^{N} \psi_i(s)} \tag{27}$$

where $\psi_i = \exp(-h_i(s-c_i)^2)$ is the Gaussian kernel with center c_i and width h_i. This function depends on a phase variable s which monotonically changes from 1 to 0 by (28).

$$\tau\,\dot{s} = -\alpha\, s \tag{28}$$

where α is a predefined constant. The learning of this dynamical system is the same as the previous one but x and v in (26) are replaced by demonstrated trajectories and their velocities, respectively. For obstacle avoidance, a coupling term $\mathbf{p}(\mathbf{x},\mathbf{v})$ is added to the differential equations of motion:

$$\tau\dot{\mathbf{v}} = \mathbf{K}(\mathbf{g}-\mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g}-\mathbf{x_0})\,s + \mathbf{K}\,\mathbf{f}(s) + \mathbf{p}(\mathbf{x},\mathbf{v}) \tag{29}$$

More details for this modulated dynamical system are provided in [34].

Billard and her colleagues tried to take advantage of dynamical systems by modulating them with generalized trajectories encoded in GMMs and retrieved from GMR. For example, in [46], a weighted composition of velocity of the generalized trajectory retrieved from GMR $\dot{\xi}^m(t)$ and velocity of a goal-directed trajectory extracted from a dynamical system $\dot{\xi}(t)$ is defined to make the desired trajectory.

$$\dot{\xi}^d(t) = \big(1-\gamma(t)\big)\dot{\xi}(t) + \gamma(t)\dot{\xi}^m(t) \tag{30}$$

Where $\gamma(t)$ is given by (31) which changes from 1 to 0 to during the movement (T is duration of the movement).

$$\gamma(t) = ((T-t)/T)^2 \tag{31}$$

The dynamical system to attain $\dot{\xi}(t)$ is the spring-and-damper system explained before with small modulation known as VITE model of human reaching movement [47].

$$\begin{aligned}\ddot{\xi}(t) &= \alpha\left(-\dot{\xi}(t) + \beta(\xi^g - \xi^s)\right)\\ \dot{\xi}(t) &= \dot{\xi}(t)^{\,s} + \ddot{\xi}(t)\,\Delta t\end{aligned} \tag{32}$$

where $\alpha, \beta \in R_{[0,1]}$, ξ^g is the target position, ξ^s is the current position of the robot, and $\dot{\xi}(t)$ and $\ddot{\xi}(t)$ are the current speed and acceleration of the dynamical systems.

Billard and her colleagues, in another work [42], improved the above approach with solving an optimization problem. Firstly, they found a generalized signal with GMR to model velocity of demonstrated trajectory as a function of time $\dot{x}^m(t)$, and then employ it to modulate the dynamical system in (33).

$$\ddot{\theta}^s(t) = \alpha\left(-\dot{\theta}(t) + \beta(\theta^g - \theta)\right) \tag{33}$$

where $\theta \in R^n$ is the vector of joint angles for the robot. The modulation was performed by solving the optimal control problem

$$\dot{\theta}^* = \mathrm{argmin}_{\dot{\theta}}(1-\gamma)\big(\dot{\theta}-\dot{\theta}^s\big)^T\overline{W}_\theta\big(\dot{\theta}-\dot{\theta}^s\big) + \gamma(\dot{\mathrm{x}}-\dot{\mathrm{x}}^{\mathrm{m}})^{\mathrm{T}}\overline{\mathrm{W}}_\theta(\dot{\mathrm{x}}-\dot{\mathrm{x}}^{\mathrm{m}}) \tag{34}$$

$$s.t. \quad \dot{X} = \mathrm{J}\,\dot{\theta}$$

where γ is to weight the modulation and defined by this differential equation in which $\alpha_\gamma \in R_{[0,1]}$ is a predefined scalar.

$$\dot{\gamma} = \alpha_\gamma\big(-\dot{\gamma} - \tfrac{1}{4}\alpha_\gamma\gamma\big), \quad \gamma_0 = 1 \tag{35}$$

The solution to the minimization problem and more details are explained in [42].

The works presented above were more involved with discrete movements. Here, we go through the methods dealing with periodic ones. First, we explain adaptive frequency oscillators that can be used in CPG programming or dynamical systems modulation for reproducing rhythmic waveforms.

Oscillators have important role in modeling and control of nonlinear systems. Specially, they are interesting due to their synchronization capabilities, either with other oscillators or external signals [48]. In [48], Righetti et al. proposed an algorithm that enables many kinds of oscillators to learn frequency of any periodic input without any signal processing. In their formulation, an oscillator perturbed by periodic signal F is described by general equations

$$\begin{aligned}\dot{x} &= f(x,y,\omega) + \epsilon\,F\\ \dot{y} &= f(x,y,\omega)\end{aligned} \tag{36}$$

where ω is a parameter influencing the frequency of oscillation. For this parameter, the follwing learning rule is introduced:

$$\dot{\omega} = \mp\epsilon\,F\,\frac{y}{\sqrt{x^2+y^2}} \tag{37}$$

where the sign depends on the direction of rotation of the limit cycle in x-y phase space. It is proved in [48] that this learning rule can adapt frequency of oscillation to frequency of input signal F. Note that if more than one frequency is existing, the attracting frequency component depends on its distance to the intrinsic frequency of oscillator and its intensity.

In [44], Righetti et al. investigated application of frequency adaptive oscillators for the purpose of CPG programming. They employed a coupling of adaptive Hopf oscillators to implement a kind of dynamic Fourier series representation of a periodic signal. The CPG architecture is depicted in Fig. 5 and differential equations describing the CPG are shown in (38)-(45).

It is shown that, for each frequency component of the training signal, an oscillator is considered. The outputs of the oscillators are weighted with a variable representing the amplitude of each component, and finally they are summed to make the output of CPG. With the use of negative feedback mechanism, we can learn other frequency components that haven't been learned yet. In (40) and (41), learning rules for frequency component (which is the learning rule explained above) and amplitude of them (which is base on steepest descend algorithm) are shown, respectively. For each oscillator, the phase difference between that oscillator and the first oscillator (ϕ_i) is encoded into the equations so any phase relation between oscillators can be reproduced. After convergence of the network, $F(t)$ is set to zero for regeneration of the rhythmic signal. More details about implementation of this CPG and online modulation of different variables are provided in [44].

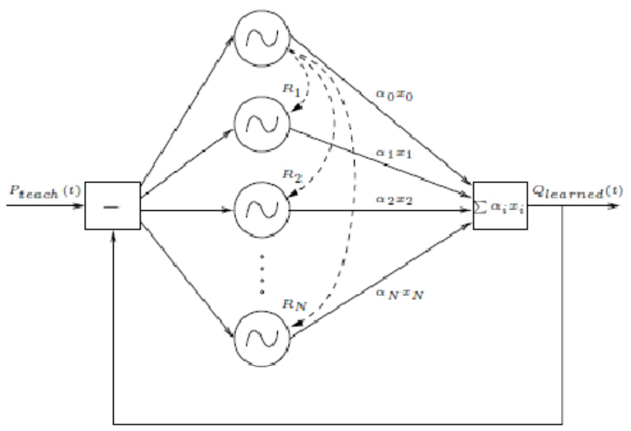

Figure 5. Structure of the network of Hofp oscillators from [44].

$$\dot{x}_i = \gamma(\mu - r_i^2)x_i - \omega_i y_i + \epsilon F(t) + \tau \sin(\theta_i - \phi_i) \quad (38)$$
$$\dot{y}_i = \gamma(\mu - r_i^2)y_i - \omega_i x_i \quad (39)$$
$$\dot{\omega}_i = -\epsilon F(t)\frac{y_i}{r_i} \quad (40)$$
$$\dot{\alpha}_i = \eta x_i F(t) \quad (41)$$
$$\dot{\omega}_i = \sin\left(\frac{\omega_0}{\omega_0}\theta_0 - \theta_i - \varphi_i\right) \quad (42)$$

with

$$\theta_i = \mathrm{sgn}(x_i)\cos^{-1}\left(-\frac{y_i}{r_i}\right) \quad (43)$$
$$F(t) = P_{teach}(t) - Q_{learned}(t) \quad (44)$$
$$Q_{learned}(t) = \sum_{i=0}^{N} \alpha_i x_i \quad (45)$$

Gams et al. [43] also used frequency adaptive oscillators for learning arbitrary periodic movements. Their network is composed of two dynamical systems. The first one (canonical dynamical system) is a composition of phase oscillators in a feedback structure with the adaptive frequency learning rule described before. The second one (output dynamical system) is the spring-and-damper system modulated by a nonlinear function of oscillation phase. The dynamics of the canonical dynamical system which is defined for each dimension of trajectory is as follows

$$\begin{aligned} \dot{\phi}_i &= \omega_i - Ke(t)\sin(\phi_i) \\ \dot{\omega}_i &= -Ke(t)\sin(\phi_i) \\ e(t) &= y_{demo}(t) - \hat{y}(t) \\ \hat{y}(t) &= \sum_{i=1}^{M} \alpha_i \cos(\phi_i) \\ \dot{\alpha}_i &= \eta \cos(\phi_i)\, e(t) \end{aligned} \quad (46)$$

where ϕ is the phase, e(t) is the input into the oscillators, $y_{demo}(t)$ is the input signal into the system, M is the number of oscillators, K is the coupling strength or constant, and η is the learning rate. From this dynamical system, the lowest non-zero frequency Ω and the phase of oscillator at that frequency are extracted and feed to the output dynamical system. The dynamics of this system is given by

$$\begin{aligned} \frac{1}{\Omega}\dot{z} &= \alpha_z(\beta_z(g - y) - z)\frac{\sum_{i=1}^{N}\psi_i w_i r}{\sum_{i=1}^{N}\psi_i}, \\ \dot{y} &= \Omega\, z, \end{aligned} \quad (47)$$

where $\psi_i = \exp(h_i(\cos(\phi_i - c_i) - 1)$, g is the anchor for the oscillatory trajectory, α_z and β_z are positive constants and r is amplitude control parameter. In the learning stage g can be set to zero and r to 1 while in time of regeneration, they can be changed for online modulation of the learned trajectory.

For learning the teaching trajectory, y is replaced by y_{demo} and weights of the nonlinear term are identified by incremental locally weighted regression (ILWR). The structure of the whole system is demonstrated in Fig. 6.

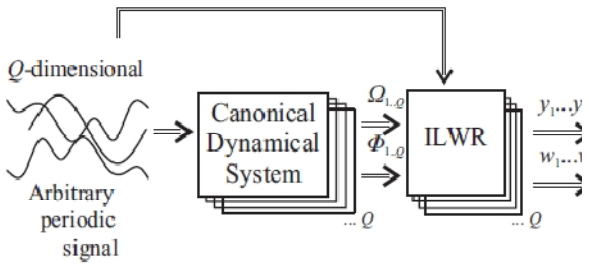

Figure 6. Structure of the network of Hofp oscillators from [43].

Ajalooeian et al. [26] proposed another approach for generation of arbitrary periodic movements with CPGs. They consider a simple base oscillator like Hopf oscillator and transform it to the desired trajectory of joint angles with a nonlinear mapping. First, a number of corresponding points on both the trajectories are selected and then associated with two multilayer perceptrons (MLP). The first MLP is a mapping from the target space of the desired trajectory to the base space of Hopf oscillator (forward mapping), and the second MLP is a mapping from the base space to the target space (backward mapping). This is due to the fact that the forward mapping might not be invisible. After learning, for regeneration of the training trajectory, the forward mapping takes the point to the base space where the point is changed with Hopf oscillator for one step time. Next, the new point is transported to the target space with backward mapping. This preliminary model is very sensitive to estimation error, so they revised the model with a small change in backward mapping. In the new model, backward mapping is defined from the points in base space to velocity vectors at the specified points of target space where the points are updated by these velocity vectors.

Finally, they proposed an extended model of their CPG to ensure generation of a limit cycle attractor with a desired basin of attraction. In this model, the limit cycles of both base and target space are surrounded by two margins. In the forward mapping, the points of limit cycles and the points in the margins from two spaces are associated with a MLP. The backward MLP is composed of mapping points of base space to velocity vectors of target space (like previous) in addition to points in the margins of base space to transverse velocity vectors of the margins in target space, directing to the limit cycle. It is proved that the resulting CPG has a limit cycle attractor with at least a basin of attraction surrounded by the introduced margins. The extended model of this CPG is illustrated in Fig. 7.

A comparative summary of the most recent and important methods described in this section is provided in Table V.

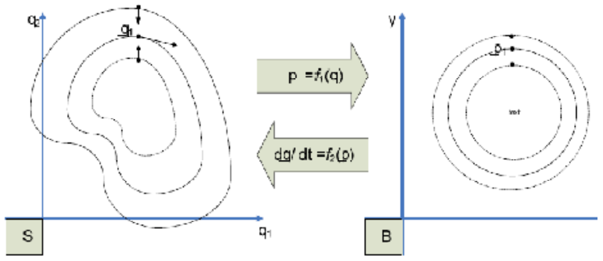

Figure 7. CPG model with Poincaré-Bendixson theorem deployed. [26].

TABLE V. COMPARISON BETWEEN SELECTED METHODS OF SKILL ENCODING BASED ON DYNAMICAL SYSTEMS

Proposed by (supervised by)	Method	Type of task	Advantages	Disadvantages	Features for recognition in functional space
M. Hersch et al. [42] (Billard)	Generalizing trajectory by GMM and GMR and integrating it into a goal-directed dynamical system with an optimal control problem	Discrete tasks	• Ability of generalization from multiple demonstrations • Goal modulation	• Need Jacobean matrix • Offline Learning	The whole trajectory obtained in joint space
P. Paster et al. [34] & Hoffmann et al. [33] (Schaal)	Training weights of basis functions in a second order dynamical system modulated with a nonlinear term	Discrete tasks	• Online Learning • Goal modulation • Spatially (initial point and goal) and temporally invariant • Obstacle avoidance from many obstacles and moving obstacles • Compress signal to a matrix of N (number of basis functions) ×Q (signal dimension)	• Number of basis functions should be determined	Weights of basis functions
Righetti et al. [44] (Ijspeert)	CPG programming with frequency adaptive oscillators and make a dynamic Fourier representation of the periodic signal	Periodic tasks	• Online learning without any optimization or regression • No preprocessing to obtain frequencies • Many Oscillators can be used • Simple modulation of frequency and amplitude	• Number of basis functions should be determined • No information about basin of attraction	The whole trajectory obtained in joint space
A. Gams et al. [43] (Ijspeert)	Using a frequency adaptive oscillator to learn the frequencies and feed them to a dynamical second order system with a nonlinear term to shape the signal	Periodic tasks	• Online learning • No preprocessing to obtain frequencies • Many oscillators can be used • Frequency, amplitude, and anchor modulation • Only needs the common frequency • Compress signal to a matrix of N×Q	• Number of frequency components should be predetemined • No information about basin of attraction	Weights of basis functions
M. Ajallooeian et al. [26] (Nili)	CPG programming by using a base oscillator and transform it to the desired signal with a nonlinear mapping	Periodic tasks	• Basin of attraction is partially known (i.e. region of attraction is determined but no information about out of that) • Any oscillator can be used • Frequency modulation • Compress signal to parameters of forward and/or backward mapping	• Structure of nonlinear mappings (e.g. MLPs) should be identified • Offline learning • Need preprocessing to obtain frequency • Amplitude modulation might be impossible	Parameters of forward and/or backward mapping (e.g. weights of MLPs)

IV. HIGH LEVEL LEARNING

High level learning is associated with goal-directed imitation or program-level imitation. It is different from low level imitation in the fact that it isn't preoccupied with pure imitation of motor or end-effector signals, but it tries to understand the task and imitate the task goals and subgoals. According to [3] high level learning decomposes a skill to sequence of action-perception units which is called symbolic encoding, and a common way for this process is to segment and encode the task as a sequence of predefined actions. However, in this study we are interested to generalize this statement to say that high level learning is seeking for a sequence of concepts. Concept is a more general term which is ubiquitous in machine learning literature and can also provide us with techniques of automatic concept generation.

Generally, the concepts are actions, behaviors, gestures, configurations, relations, etc. which might be encoded in flat or hierarchical graphs and learning can be conducted via topology construction, rule extraction, etc. For example in [], human motions are encoded as predefined postures with different levels of granularity for symbolic representation of the motion using hidden Markov models (HMM) and then correspondence problem is solved by this symbolic representation.

Lopes et al. in [29], uses two concepts of grasp and release (although they don't name them concepts) as transition phases between two predefined general states of rest and move to symbolically represent objects motion in a program-level imitation task of a robot. Grasping of an object is detected when it starts to move or it is occluded by hand. Releasing is identified by detecting that the grasped object becomes static while hand is moving. The proposed algorithm marks every grasping and releasing in the trajectory of objects and the desire features of world are recorded in the states obtained.

The works described in the following paragraphs push more emphasis on conceptual framework of high level imitation learning.

Schaal and their colleagues in their works [10], [49] proposed a conceptual imitation learning system based on sensory-motor representation which comprises a library of movement primitives. Movement primitives are sequence of actions that fulfill a completed goal behavior [10]. They can be as simple as single actions, but for the best performance in real problems, they are represented in complete temporal behaviors like “grasping a cup” [3]. Due to our interests to the term of concept described previously, we can say that these movement primitives are also concepts. Overall structure of this system is demonstrated in Fig. 8.

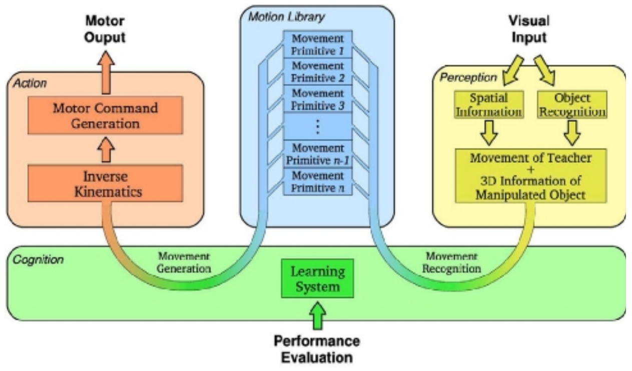

Figure 8. Conceptual structure of imitation learning system based on sensory motor representations from [34].

According to the proposed framework, visual sensory information is transformed into information about objects and their spatial location, so the posture of the teacher and position of the objects are derived. Subsequently, this information invokes a set of movement primitives and the demonstrated actions are performed by motor system of the robot. In this scheme, the mapping from sensory information to correct movement primitives is an important issue which is constructed in an assimilation phase described in [50]. Shortly, there is a dual-rout process with an active and inactive mode. If the robot faces a learned movement which is available in the library of movement primitives, the active mode is invoked and the appropriate movement primitive is regenerated and also refined by the teacher demonstration. On the other hand, if the robot faces a new movement, it goes to inactive mode and tries to learn that movement by low-level imitation, and also add it the library. Note that in this general framework, after the initial imitation, self-improvements techniques can be employed by performance evaluation criterion to have more efficient motor commands [3].

Another conceptual framework is proposed by Mobahi et al. [11] which is somehow similar to the above-mention system. However, it is directly going to make concepts out of observations, and also it has two types of memory: working memory and long-term memory. In this scheme, like the previous one, the agent is living in two functional modes of being interactive and non-interactive. In interactive mode, it interacts with the teacher to learn the relations between sensory inputs and concepts in the memory. Actually, in this mode, the agent refines its knowledge by reinforcement signals of the teacher, e.g. finding weights of relations between states to concepts in working memory or transferring the concepts from working (long-term) memory to long-term (working) memory. In non-interactive mode, it tries to regenerate the movement with its observation using babbling without further interaction with the teacher. In fact, when there is no match for the demonstrated action in the conceptual memory of the agent, it goes to non-interactive mode and tries to make it. Pseudocode for the proposed algorithm is provided in Fig. 9.

```
$\mathbf{x} := \text{Sense}()$
$i := \underset{\mathbf{x}_k \in W}{ArgMin} \|\mathbf{x} - \mathbf{x}_k\|$
$j := \underset{\mathbf{x}_k \in L}{ArgMin} \|\mathbf{x} - \mathbf{x}_k\|$
$if \|\mathbf{x} - \mathbf{x}_i\| < \|\mathbf{x} - \mathbf{x}_j\|$
    $then \quad k := \underset{p \in C}{ArgMax}\, \mathbf{W}_{ip}, \mathbf{y} := \mathbf{y}_k$
    $else \quad k := c_j, \mathbf{y} := \mathbf{y}_k$
$\text{Perform}(\mathbf{y})$
$R := \text{Get\_Reinforcement}()$
$if (R > 0, \|\mathbf{x} - \mathbf{x}_i\| < \|\mathbf{x} - \mathbf{x}_j\|)$
              $p := \text{New\_L}(),$
    $then$    $\mathbf{x}_p := \mathbf{x}_i, c_p := k,$
              $\text{Delete\_W}(i)$
$if (R > 0, \|\mathbf{x} - \mathbf{x}_i\| \geq \|\mathbf{x} - \mathbf{x}_j\|)$
    $then \quad \Delta\mathbf{x}_j := \eta(\mathbf{x} - \mathbf{x}_j)$
$if (R < 0, \|\mathbf{x} - \mathbf{x}_i\| \geq \|\mathbf{x} - \mathbf{x}_j\|, \|\mathbf{x} - \mathbf{x}_i\| > G)$
              $p := \text{New\_W}(), \boldsymbol{\mu}_p := \mathbf{y}, \mathbf{x}_p := \mathbf{x},$
    $then$    $\underset{q \in C}{\mathbf{W}_{pq}} := \exp(-\|\mathbf{y}_q - \boldsymbol{\mu}_p\|^2)$
$if (R < 0, \|\mathbf{x} - \mathbf{x}_i\| < \|\mathbf{x} - \mathbf{x}_j\|, \|\mathbf{x} - \mathbf{x}_i\| < F)$
              $\mathbf{W}_{ik} := 0$
    $then$    $if (\forall \mathbf{y}_k \in M, \mathbf{W}_{ik} = 0)$
                        $\text{Find } \mathbf{y}^* \text{ such that Perform}(\mathbf{y}^*) = \mathbf{x},$
                        $q := \text{New\_C}()$
                        $\mathbf{y}_q := \mathbf{y}^*$
              $then$    $p := \text{New\_L}(),$
                        $\mathbf{x}_p := \mathbf{x}, c_p := q$
                        $\text{Delete\_W}(i)$
                        $\forall \mathbf{x}_p \in W : \mathbf{W}_{pq} = \exp(-\|\mathbf{y}_q - \boldsymbol{\mu}_p\|^2)$
```

Figure 9. Pseudocode for concept-oriented imitation in [11].

In [40], Chella et al. presents a cognitive framework for imitation learning. The proposed framework has three computational areas: subconceptual area, conceptual area, and linguistic area (cf. Fig. 10). In subconceptual area, low-level processing of perceptual data like color segmentation, feature extraction, hand recognition, key-frame generation, scene reconstruction, and on other hand motor control of the system is performed.

The conceptual area is composed of perceptual space and action space. The perceptual space is related to the objects, so it is given by shape and color of the objects. Each point in this space is also assigned to a unique identifier with spatial information. Indeed, this space is used to assign high level terms to the objects like square, circle, green, red, etc. Action space is related to the actions and tries to encode the trajectories. For encoding, PCA is employed and projection coefficients in () described in the previous section are selected as features to represent the trajectory. This space is used to generalize trajectories and assign symbolic names to actions.

Next, natural concepts are attained form data provided in perceptual space and action space by clustering. It means that each cluster is a concept which represents a high level term like circle, red, etc. in the perceptual space and reach, grasp, etc. in the action space. Finally, imitation of the whole task is accomplished by sequencing and combining of obtained concepts. This can be performed by symbolic reasoning in the linguistic area for which some approaches are explained in the beginning of this section. It is worth noting that temporal representation and sequencing of the concepts derived in the two previously reviewed conceptual frameworks by Billard and Mobahi can also be accomplished by symbolic reasoning (although they haven't declared in their publications).

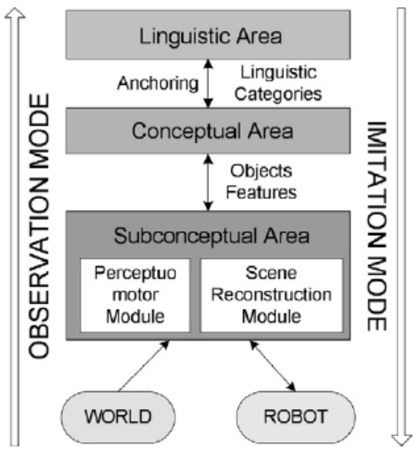

Figure 10. Computational areas of the proposed architecture in [40].

Amirpour et al. [51] also proposed a conceptual model for imitation learning. In this approach a number of operators are fed to the agent mind as predefined knowledge, and agent exploits these operators to derive relations between sensory information which make concepts of the task. In addition, the temporal sequence of concepts is represented with a fully consistent timing graph of the task.

The algorithm starts with perception, so the sensory information from multiple experiments is observed and mathematical and logic operators are given by designer. Then Concepts should be extracted. First, all mathematical relations between sensory inputs are calculated with a feed forward network. Then, all common subsequences within all experiments are detected and each common subsequence is introduced as a new concept. The next step is to derive scenarios. First, a general timing graph is formed out of all extracted concepts. Then all fully consistent subgraphs (i.e., graphs in which temporal sequence of all concepts are the same through all the experiments) are found and temporal sequence of concepts in each one is arranged, and the final scenarios are obtained in this way. As there might be more than one plausible scenarios, they are evaluated by the teacher and reward or punishment is received from the teacher which is used to update probability of selecting each scenario.

V. Conclusion

We presented a survey on imitation learning for robot programming and control. Some computational approaches and methods for imitation learning were summarized. Specially, learning inverse kinematics, low level learning, and high level learning were highlighted with more explanations and details. It seems that the topic has been investigated in different aspects during the past decade and many of the questions and challenges have been answered. However, progress in devising a general, fast, and user friendly framework for online imitation learning which is suited for both high level and low level skills and makes applicability of complex robots in human and artifact societies pervasive is continuing and more endeavors are needed.

More specifically, online motor control of robots with complicated ingredients and models, controlling interaction of robots in social environments (including the questions of who to imitate and when to imitate), enhancing ability of robots to understand intention of the teachers, reducing number of demonstrations for teaching and generalization, reducing amount of prior knowledge as much as it doesn't spoil accuracy or speed of learning, providing robots with ability of explorative or developmental self-learning or interactive learning like reinforcement learning, and incorporating ability of abstract knowledge transfer between learned skills and semi-automatic behavioral switching between the skills are critical issues which can be scrutinized.

References

[1] M. A. Wood and J. J. Bryson, Skill Acquisition Through Program-Level Imitation in a Real-Time Domain, IEEE Trans. Syst., Man, Cybern., vol. 37, no. 2, april 2007.

[2] J. Call and M. Carpenter, "Three sources of information in social learning," Imitation in animals and artifacts, MIT Press, 2002 ,pp. 211-228.

[3] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot Programming by Demonstration, Handbook of Robotics, Chapter 59, Sept. 2007.

[4] S. Schaal, A. J. Ijspeert, and A. Billard, "Computational Approaches to Motor Learning by Imitation," in Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences, 2003.

[5] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillmann. Robot programming by demonstration (RPD) - Using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. In Proceedings of the International Symposium on Industrial Robots (ISIR), pages 685-693, 1994.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Transactions on Robotics and Automation, 10(6):799-822, 1994.

[7] C.P. Tung and A.C. Kak. Automatic learning of assembly task using a DataGlove system. In IIEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 1, pages 1-8, 1995.

[8] K. Ikeuchi and T. Suchiro. Towards an assembly plan from observation, part I: Assembly task recognition using face-contact relations (polyhedral objects). In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 3, pages 2171-2177, May 1992.

[9] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation, 37(2):286-298, 2007.

[10] S. Schaal. Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences, 3(6):233-242, 1999.

[11] H. Mobahi, M. Nili Ahmadabadi, and B. N. Araabi, (2007) “A biologically inspired method for conceptual imitation using reinforcement learning”, Applied Artificial Intelligence, 21:3, 155 - 183

[12] A. Billard, M. Mataric, Learning human arm movements by imitationL evaluation of a biologically inspired connectionist architecture

[13] E. Oztop, M. Kawato, and M.A. Arbib. Mirror neurons and imitation: A computationally guided review. Neural Networks, 19(3):254-21, 2006.

[14] E.L. Sauser and A.G. Billard. Dynamic updating of distributed neural representations using forward models. Biological Cybernetics, 95(6):567-588, 2006.

[15] A. Billard and G. Hayes. Drama, a connectionist architecture for control and learning in autonomous robots. Adaptive Behavior, 7(1):35-64, 1999.

[16] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 3, pages 2700-2705, April 1996.

[17] R. Dillmann, M. Kaiser, and A. Ude. Acquisition of elementary robot skills from human demonstration. In Proceedings of the International Symposium on Intelligent Robotic Systems (SIRS), pages 1-38, July 1995.

[18] J. Yang, Y. Xu, and C.S. Chen. Hidden Markov model approach to skill learning and its application in telerobotics. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 1, pages 396-402, May 1993.

[19] H. Bekkering and W. Prinz, “Goal representations in imitative actions,” Imitation in animals and artifacts, MIT Press, 2002, pp. 555-572.

[20] K. Dautenhahn and C.L. Nehaniv, “The agent-based perspective on imitation,” Imitation in animals and artifacts, MIT Press, 2002, pp. 1-40.

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

[22] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, “Operational space control: a theoretical and emprical comparison,” International Journal of Robotics Research, vol. 27, pp. 737–757, 2008.

[23] A. D'Souza, S. Vijayakumar, S. Schaal, Learning inverse kinematics. In Proceedings of the IEEE/RSJ International Conference on Intelligence in Robotics and Autonomous Systems (IROS 2001). Maui, HI, USA, Oct 2001.

[24] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on Man-Machine Systems, 10(2):47-53, 1969.

[25] M. Lopes and J. Santos-Victor, “Visual learning by imitation with motor representations,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 35, no. 3, pp. 438–449, Jun. 2005.

[26] M. Ajallooeian, M. Nili Ahmadabadi, B. N. Araabi, H. Moradi, An Imitation Model based on CPG with application to Robotic Marionette Behavior Learning, in Proc. In IIEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[27] Aris Alissandrakis Chrystopher L. Nehaniv Kerstin Dautenhahn, Solving the Correspondence Problem Between Dissimilarly Embodied Robotic Arms Using the ALICE Imitation Mechanism, 2003.

[28] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, “Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments,” IEEE Trans. Syst., Man, Cybern. A, Syst. Humans, vol. 32, no. 4, pp. 482–496, Jul. 2002.

[29] M. Lopes, and J. Santos-Victor, A Developmental Roadmap for Learning by Imitation in Robots, IEEE Trans. on System, Man, and Cybernetics, vol. 37, no. 2, April 2007.

[30] R. Fletcher, Practical Methods of Optimization, 2nd ed. Chichester, U.K.: Wiley, 1987.

[31] K. Hosoda and M. Asada, “Versatile visual servoing without knowledge of true Jacobian,” in Proc. Int. Conf. Intell. Robots and Syst., Munchen, Germany, Sep. 1994, pp. 186–193.

[32] N. Mansard, M. Lopes, J. Santos-Victor, and F. Chaummette, “Jacobian learning methods for sequencing visual servoing,” in Proc. IEEE/RSJ Int. Conf. IROS, 2006, pp. 4284–4290.

[33] H. Hoffmann, P. Pastor, D. Park, and S. Schaal, “Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance,” in Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009.

[34] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, Learning and Generalization of Motor Skills by Learning from Demonstration, in Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009.

[35] H. Cruse, et al. (1990). On the cost functions for the control of the human arm movement. Biological Cybernetics, 62(6): 519–28.

[36] N. Delson and H.West. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 30-36, 1996.

[37] N. Delson and H.West. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 30–36, 1996.

[38] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Extraction of essential interactions through multiple observations of human demonstrations. IEEE Transactions on Industrial Electronics, 50(4):667-675, 2003.

[39] J. Yang, Y. Xu, and C.S. Chen. Human action learning via hidden Markov model. IEEE Transactions on Systems, Man, and Cybernetics-PartA: Systems and Humans, 27(1):34–44, 1997.

[40] A. Chella, H. Dindo, and I. Infantino. A cognitive framework for imitation learning. Robotics and Autonomous Systems, 54(5):403–408, 2006.

[41] A.J. Ijspeert, Central pattern generators for locomotion control in animals and robots: a review, Neural Networks Vol 21/4 pp 642-653, 2008.

[42] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations," IEEE Transactions on Robotics, 2008.

[43] A. Gams, L. Righetti, A. J. Ijspeert and J. Lenarcic, A Dynamical System for Online Learning of Periodic Movements of Unknown Waveform and Frequency, 2008.

[44] L. Righetti, A.J. Ijspeert, “Programmable central pattern generators: an application to biped locomotion control,” Proc. IEEE International Conference on Robotics and Automation, 2006.

[45] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 1398-1403. 2002.

[46] F. Guenter, M. Hersch, S. Calinon, and A. Billard, “Reinforcement learning for imitating constrained reaching movements,” RSJ Advanced Robotics, vol. 21, no. 13, pp. 1521–1544, 2007.

[47] M. Hersch and A. Billard, A biologically-inspired model of reaching movements, in Proc. IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, Pisa, (2006).

[48] L. Righetti, J. Buchli, A.J. Ijspeert, “Dynamic hebbian learning in adaptive frequency oscillators,” Physica D, 216(2), 269-281, 2006.

[49] D. Sternad and S. Schaal. Segmentation of endpoint trajectories does not imply segmented control. Experimental Brain Research, 124/1:118-136, 1999.

[50] Y. Demiris and G. Hayes. Imitation as a dual route process featuring predictive and learning components: a biologically-plausible computational model. In K. Dautenhahn and C. Nehaniv, editors, Imitation in Animals and Artifacs, pages 327-361. MIT Press, 2002.

[51] S. Amirpour Amraii, M. Nili Ahmadabadi, and B. Nadhar Araabi, “Cocept based Framework for Learning by Imitation,” University of Tehran.

[52] S. Calinon and A. Billard. What is the teacher's role in robot programming by demonstration? Toward benchmarks for improved learning. Interaction Studies. Special Issue on Psychological Benchmarks in Human-Robot Interaction, 8(3):441-464, 2007.