VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0016

Preface

This special issue of the Journal of Cybernetics and Information Technologies (CIT) contains the proceedings from the AIMSA'2012 Workshop on Advances in Robot Learning and Human-Robot Interaction that was held on 12th of September 2012 in Varna, Bulgaria.

The workshop is organized by the IEEE Technical Committee on Robot Learning, with chairs: Petar Kormushev, Edwin Olson, Ashutosh Saxena, and Wataru Takano. The focus is on recent advances in applying machine learning and AI approaches to robotics in general and in particular to the domain of human-robot interaction. The interest in applying machine learning approaches within the robotics community is continuously growing, in response to the fact that robot hardware is progressively becoming more complex. The conventional, manually-engineered solutions to robot perception and control are reaching their limits, and there is an acute need for automated learning methods to cope with the ever-increasing complexity of robots.

This special issue contains nine selected papers, presenting results on a wide range of robotic platforms, such as robot arm manipulators, humanoid robots, and quadruped robots. Both theoretical and practical results are presented, on a variety of real-world scenarios for robot task learning and human-robot interaction. In addition, the issue contains abstracts of two invited talks, in which the invited speakers present the latest research developments in their respective fields. Assistant Professor Alexander Stoytchev, who is the Director of the Developmental Robotics Laboratory at Iowa State University, USA, presents an overview of the progress in the developmental approach to robotic intelligence, as well as the latest results in this field from his laboratory. Associate Professor Yukie Nagai, from

Osaka University, Japan, presents recent advances in human-robot interaction, with focus on how the interaction shapes the way robots learn.

We hope that the readers will find this issue stimulating and inspiring further research.

We would like to cordially thank the local organizer of the workshop, which is the Institute of Information and Communication Technologies (IICT), Bulgarian Academy of Sciences, and especially to Associate Professor Gennady Agre, for the support and guidance throughout the preparation of this workshop.

Petar Kormushev, Italian Institute of Technology, Italy

Edwin Olson, University of Michigan, USA

Ashutosh Saxena, Cornell University, USA

Wataru Takano, University of Tokyo, Japan

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0017

Which Object Comes Next?
Grounded Order Completion by a Humanoid Robot

C. Schenck, J. Sinapov, A. Stoytchev

Developmental Robotics Laboratory Iowa State University, Ames IA, USA
Emails: cschenck@iastate.edu jsinapov@iastate.edu alexs@iastate.edu

Abstract: *This paper describes a framework that a robot can use to complete the ordering of a set of objects. Given two sets of objects, an ordered set and an unordered set, the robot's task is to select one object from the unordered set that best completes the ordering in the ordered set. In our experiments, the robot interacted with each object using a set of exploratory behaviors, while recording feedback from two sensory modalities (audio and proprioception). For each behavior and modality combination, the robot used the feedback sequence to estimate the perceptual distance for every pair of objects. The estimated object distance features were subsequently used to solve ordering tasks. The framework was tested on object completion tasks in which the objects varied by weight, compliance, and height. The robot was able to solve all of these tasks with a high degree of accuracy.*

Keywords: *Developmental robotics, object exploration, grounding.*

1. Introduction

Humans can detect order in an unordered set of objects at a very early age. Ordering tasks frequently appear on modern intelligence tests [7, 8]. They are also tightly integrated in many educational methodologies. For example, in the Montessori method [12], a 100-year-old method of schooling for children that has been shown to outperform standard methods [10, 11], children are encouraged to solve different object ordering tasks with specialized toys [16]. These strongly suggest that the ability to discover orderings among sets of objects is an important skill. Indeed, studies in

psychology have revealed that this skill is learned at a very early age [22, 6, 2, 3].

Because order completion skills are so important for humans they should be important for robots that operate in human environments as well. Previous research has shown that robots can successfully form object categories [15, 14, 13, 24] and solve the odd-one-out task [19]. Object ordering tasks, however, have not received a lot of attention from the robotics community to date.

This paper proposes a method for discovering orderings among groups of objects. The experiments were conducted with an upper-torso humanoid robot, which interacted with the set of objects using a set of stereotyped exploratory behaviors. The robot recorded both auditory and proprioceptive data during each interaction and then extracted features from the sensory records. Using the extracted features for each object, the robot was able to estimate a pairwise distance matrix between every pair of objects. Then given three objects that form an ordered set, the robot's model was queried to pick one object from another group of four to complete the ordering in the first set. The results show that the robot was able to pick the correct object that completes the ordering with a high degree of accuracy and that different exploratory behaviors and sensory modalities are required to capture different ordering concepts.

2. Related work

Object ordering tasks appear on multiple intelligence tests. For example, on the Intelligence and Development Scales (IDS) test [7], children are asked to sort lines of varying length. In a more common test, the Wechsler Intelligence Scale for Children (WISC) [8], participants are asked to place images from a story into a logical sequence. While it is not currently feasible for a robot to understand the events taking place in an image, these two tests show that, given an understanding of the objects, knowledge of how to order them is a strong indicator of intelligence.

Several studies have shown that young children have a fundamental understanding of the concepts underlying ordering. G r a h a m *et al.* [6] found that children between the ages of 2 and $4\frac{1}{2}$ can easily judge an object as "big" or "small" when compared to another object. Two studies by E b e l i n g and G e l m a n [2, 3] found similar results. Interestingly, all three studies found that children were much better able to judge an object as "big" or "small" when compared with immediately viewable objects as opposed to making the judgment based on the object's absolute size [6], its normative size (i.e., how big it is compared to the typical object in the category) [2], or its functional size (i.e., how big it is in relation to the function it is to perform) [3]. The ability to compare an object to other directly viewable objects is a prerequisite for successfully performing the task of ordering and since it is present more strongly than other types of comparisons (absolute, normative, or functional) at such an early age, it must be fundamental to intelligence.

In another study with 1 to 3-year-olds, S u g a r m a n [22] found that the order in which children interact with objects tends to be influenced by the class and perceptual similarity of the current object to the previously explored object. Additionally, it was observed that the older children relied less on the class of the object to pick the

(a) Weight Cylinders (b) Pressure Cylinders (c) Cones and Noodles

Fig. 1. The three sets of objects used in the experiments

next object and more on perceptual similarity. This paper uses a similar method to determine orderings. The perceptual distances between each pair of objects is used to determine the best object to complete the ordering.

In machine learning, the problem of ranking (i.e., placing a set of data in the correct order) has been well studied [1]. There are many algorithms that can solve ranking with a high degree of accuracy. It is difficult to use standard ranking methods, however, to perform order completion tasks, especially when the number of objects is small. Additionally, standard ranking methods are often supervised or semi-supervised. On the other hand, the method proposed in this paper solves the task in both unsupervised and supervised settings.

Not a lot of research has been done in robotics on discovering orderings in small groups of objects. Measuring the similarity between objects, however, is a common way to solve tasks in robotics. There have been numerous experiments that have demonstrated robots' ability to measure perceptual as well as functional object similarity for varying tasks [15, 14, 13, 24, 23, 18, 4, 19]. Multiple studies [13, 17, 20, 18] have used the similarity of perceptual features to categorize objects in an unsupervised manner. In [19], perceptual distances between objects were used to solve the odd-one-out task. This paper builds on this previous work by proposing a method to solve the order completion task.

3. Experimental platform

All experiments were performed with the lab's upper-torso humanoid robot, which has two 7-DOF Barrett Whole Arm Manipulators (WAMs) as its actuators, each with an attached Barrett Hand. The robot captured proprioceptive information from the built-in sensors in the WAM that measure the angles and the torques applied to each joint at 500 Hz. The robot also captured audio data through an Audio-Technica U853AW cardioid microphone mounted in its head at the standard 16-bit/44.1 kHz over a single channel.

The robot was tested on three ordering concepts: ordering by *weight*, ordering by *compliance*, and ordering by *height*. Fig. 1 shows the three sets of objects that were used in the experiments. The first two are standard Montessori toys. The *weight cylinders* are composed of six pairs of objects (for a total of twelve objects) that vary by weight, with the objects in each pair having the same weight. All the *weight cylinders* are functionally identical except for their weight. The *pressure cylinders* are composed in a similar manner (six pairs of objects) except that they vary by the

amount of pressure required to depress the rod on top of the object. The *cones* and *noodles* are composed of five green, styrofoam cones of varying sizes and five pink, foam pieces (cut from a water noodle) ranging in size from small to large. Because the object's in the first two sets are visually identical, this task cannot be solved with vision alone. In fact, the robot did not use vision at all to solve the ordering task.

The robot performed nine behaviors on each of the objects: *grasp*, *lift*, *hold*, *shake*, *drop*, *tap*, *poke*, *push*, and *press*. Additionally, the behavior *rattle* was performed on the *weight cylinders* and the *pressure cylinders*. Each behavior was encoded as a trajectory in joint-space for the left arm using the Barrett WAM API and executed using the default PID controller. All behaviors were performed identically on each object with the exception of *grasp* and *tap*, which were adjusted automatically based on the current visually detected location of the object. Fig. 2 shows the robot performing each behavior on one of the pressure cylinders.

At the start of each trial, the experimenter placed one of the objects on the table in front of the robot. The robot then performed the exploratory behaviors on the object, with the experimenter placing the object back on the table if it fell off. This was repeated five times for each of the *cones* and *noodles* and ten times for the rest of the objects. The data for the *cones* and *noodles* was collected at an earlier time than for the rest of the objects, which is why only five repetitions were done and the behavior *rattle* was not performed on them. During each behavior, the robot recorded proprioceptive data in the form of joint torques applied to the arm over time and auditory data in the form of a wave file. Visual input was used only to determine the location of the object for the *grasp* and *tap* behaviors.

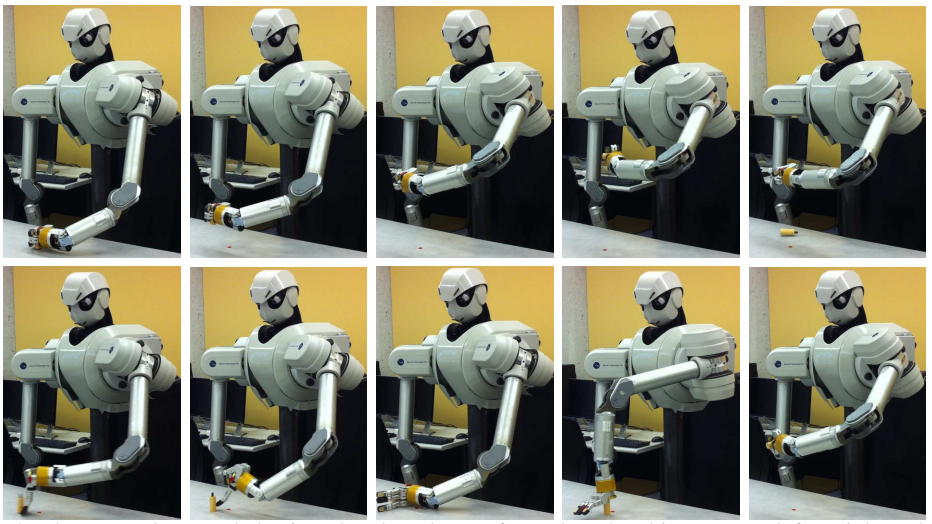


Fig. 2. The ten exploratory behaviors that the robot performed on the objects. From left to right and top to bottom: *grasp*, *lift*, *hold*, *shake*, *drop*, *tap*, *poke*, *push*, *press*, and *rattle*. The *rattle* behavior wasn't performed on the *cones* and *noodles*. The object in this figure is one of the pressure cylinders. After some of the behaviors (e.g., drop), the object was moved back to the red marker location on the table by the experimenter.

4. Feature extraction

4.1. Sensorimotor feature extraction

The auditory feedback from each behavior was represented as the Discrete Fourier Transform (DFT) of the sound's waveform, computed using 33 frequency bins. Thus, each interaction produced a $33 \times n$ matrix, where each column represented the intensities for different frequencies at a given point in time (i.e., n was the number of samples). The DFT matrix was further discretized uniformly into 10 temporal bins and 10 frequency bins. Thus, the auditory feature vector for each interaction was a $10 \times 10 = 100$ dimensional real-valued vector.

The proprioceptive feedback was represented as 7 time series of detected joint-torques, one for each of the robot's joints. To reduce the dimensionality of the data, each of the series was uniformly discretized into 10 temporal bins. Thus, the proprioceptive features for each interaction were represented by a $7 \times 10 = 70$ dimensional real-valued vector. As described next, the computed auditory and proprioceptive features were used to estimate the pairwise distances for each pair of objects.

4.2. Object feature extraction

Let $\mathcal{C}$ be the set of sensorimotor contexts, i.e., each $c \in \mathcal{C}$ corresponds to a behavior-modality combination (e.g., audio-shake), and let O denote the full set of objects. The goal of the object feature extraction routine is to compute a distance matrix $\mathbf{W}^c$ such that each entry $W^c_{ij} \in \mathbb{R}$ encodes how perceptually different objects o_i and o_j are in sensorimotor context c. Let the set $\mathcal{X}^c_i = [x_1, ..., x_D]^c_i$ contain the sensorimotor feature vectors detected for each of the D exploratory trials with object o_i in context c. The distance between two objects o_i and o_j in context c can be represented by the expected distance between the feature vectors in $\mathcal{X}^c_i$ and the feature vectors in $\mathcal{X}^c_j$, i.e.,

$$W^c_{ij} = \mathbf{E}[d_{L2}(x_a, x_b) | x_a \in \mathcal{X}^c_i, x_b \in \mathcal{X}^c_j],$$

where d_{L2} is the L2-norm distance function. This expectation is estimated by:

$$W^c_{ij} = \frac{1}{|\mathcal{X}^i_c| \times |\mathcal{X}^j_c|} \sum_{x_a \in \mathcal{X}^i_c} \sum_{x_b \in \mathcal{X}^j_c} d_{L2}(x_a, x_b).$$

The result is a set $\mathcal{W}$ of object distance matrices, where each $\mathbf{W}^c \in \mathcal{W}$ encodes the pairwise perceptual distance for each pair of objects in O. The next section describes how these matrices can be used to decide which one of a given set of objects best completes a given order.

5. Methodology

5.1. Problem formulation

Each order completion task is formulated as follows. Let O denote the set of objects explored by the robot. Let $\mathcal{L}$ denote an *ordered subset* of O, i.e., $\mathcal{L} = o_1, o_2, \ldots, o_N$ where each $o_i \in O$. Furthermore, let $\mathcal{G} \subset O$ be an unordered set of M objects denoting the set of candidate objects that could be selected to complete the order. Finally, let

$\mathcal{W}$ be a set of distance matrices such that for a given sensorimotor context c, the $|O| \times |O|$ matrix $\mathbf{W}^c \in \mathcal{W}$ encodes the pairwise object distances in that context.

In this setting, the task of the robot's model is to select one object from $\mathcal{G}$ that correctly completes the order specified by the ordered set $\mathcal{L}$. The idea behind the approach presented here is to define an objective function that can evaluate the quality of a proposed order and use that function to select an object from the set $\mathcal{G}$. The next sub-section describes the objective function as well as how that function is used to pick an object that completes the order.

5.2. Selecting the best order completion candidate

Let $q(\mathcal{L}, \mathbf{W}^c)$ denote the objective function that measures the quality of the order $\mathcal{L}$ with respect to the matrix $\mathbf{W}^c$. That function is defined as:

$$q(\mathcal{L}, \mathbf{W}^c) = \sum_{o_i \in \mathcal{L}} \sum_{o_j \in \mathcal{L}} \left(W^c_{ij} - d(o_i, o_j, \mathcal{L}) \right)^2,$$

where the function d is defined as

$$d(o_i, o_j, \mathcal{L}) = \sum_{r = o_i \ldots o_{(j-1)} \in \mathcal{L}} W^c_{r(r+1)}.$$

In other words, the function d approximates the distance between objects o_i and o_j by summing up the distances between adjacent elements in the ordered set $\mathcal{L}$. Thus, the function q measures the squared difference between the true distance matrix and the one approximated by the proposed ordering. It is used by the robot's model to complete a given ordered set of objects as follows. For each object o_k from the unordered set $\mathcal{G}$, let $\{\mathcal{L}, o_k\}$ denote the *ordered set* of objects produced by adding object o_k to the end of the ordered set $\mathcal{L}$. In this setting, the model selects the object o_k that maximizes the objective function $q(\{\mathcal{L}, o_k\}, \mathbf{W}^c)$.

5.3. Order completion using multiple sensorimotor contexts

The method presented so far can only use one distance matrix $\mathbf{W}^c$ that is specific to one sensorimotor context c. For many tasks, however, it may be desirable to use multiple sources of information about how objects relate to each other. For example, if the given ordered set of objects $\mathcal{L}$ is ordered by weight, there may be several exploratory behaviors that capture relevant proprioceptive information for solving the task (e.g., *lifting* and *holding* in place).

The set $\mathcal{W}$ contains multiple matrices encoding the pairwise object dissimilarities computed for a given set of sensorimotor contexts. For each object $o_k \in \mathcal{G}$, let the function completes$(\mathcal{L}, o_k, \mathbf{W}^c)$ return 1 if o_k is selected as the object completing the order and 0 otherwise. Given the set of all matrices $\mathcal{W}$, the ordered set $\mathcal{L}$, and the candidate set $\mathcal{G}$, the model selects the object $o_k \in \mathcal{G}$ that maximizes the following function:

$$\text{score}(o_k) = \sum_{\mathbf{W}^c \in \mathcal{W}} w_c \times \text{completes}(\mathcal{L}, o_k, \mathbf{W}^c),$$

where w_c is a weight that encodes the relevance of sensorimotor context c.

In the experiments described in the next section, three weighting methods are evaluated. Whereas everything in this paper so far has been unsupervised, two of

these weighting methods are supervised (methods 2 and 3). In the first method, the weights are *uniform*. In other words, for all c, $w_c = 1.0$.

In the second method, the weights are set to the estimated accuracy of using sensorimotor context c to solve the specific ordering task. In other words, the robot's model estimates the accuracy of a context c by running the method described in the previous subsection on a training set of tasks of the form $[\mathcal{L}, \mathcal{G}]$ for which the correct answers are known. Once the weights for all contexts have been estimated, the model uses those weights on subsequent tasks for which the answers are not known in advance.

The third method that was used to combine sensorimotor contexts is boosting. It was implemented using the AdaBoost algorithm [5]. It is briefly summarized here. Given a set of m tasks $[\mathcal{L}_1, \mathcal{G}_1], [\mathcal{L}_2, \mathcal{G}_2], ..., [\mathcal{L}_m, \mathcal{G}_m]$ for which the correct answers $o_k^1 \in \mathcal{G}_1, o_k^2 \in \mathcal{G}_2, ..., o_k^m \in \mathcal{G}_m$ are known, initialize the training weights as $D_1(i) = \frac{1}{m}$ for $i = 1, ..., m$. For each iteration $t = 1, ..., T$, select the sensorimotor context $c^*(t)$ such that $c^*(t) = \arg\min_{c \in C} \xi_c$. The error ξ_c of a context c is computed as

$$\xi_c = \sum_{i=1}^{m} D_t(i) \left[1 - \text{completes}(\mathcal{L}_i, o_k^i, \mathbf{W}^c)\right],$$

where $o_k^i \in \mathcal{G}_i$ is the object that correctly completes the ordering $\mathcal{L}_i$. Next, the parameter α_t is computed as a function of $\xi_{c^*(t)}$ as follows

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \xi_{c^*(t)}}{\xi_{c^*(t)}},$$

where $\xi_{c^*(t)}$ is the error of the selected context in iteration t. After each iteration, the training weights for all $i = 1, ..., m$ are updated as follows

$$D_{t+1}(i) = D_t(i) \exp\left[-\alpha_t (2 * \text{completes}(\mathcal{L}_i, o_k^i, \mathbf{W}^{c^*(t)}) - 1)\right],$$

where $\mathbf{W}^{c^*(t)}$ is the object distance matrix of the context selected during iteration t, and then normalized such that they sum to 1. It is worthwhile to note that the expression $-\alpha_t (2 * \text{completes}(\mathcal{L}_i, o_k^i, \mathbf{W}^{c^*(t)}) - 1)$ comes out to $+\alpha_t$ if context $c^*(t)$ incorrectly predicts the object to complete the ordering $\mathcal{L}_i$ and $-\alpha_t$ otherwise. In essence, the training weights are altered such that tasks that context $c^*(t)$ is incorrect on are weighted higher and tasks that it is correct on are weighted lower.

Finally, the weight w_c for each sensorimotor context is computed by

$$w_c = \sum_{t=1}^{T} \alpha_t [c \equiv c^*(t)],$$

where $[c \equiv c^*(t)]$ is 1 if c was chosen during iteration t and 0 otherwise. In the experiments described in this paper, T was set to 50. Results did not change significantly with a higher value for T.

5.4. Evaluation

The model was evaluated independently on each of the three ordering concepts. Fifty tasks were randomly sampled for each concept as follows: four objects were sampled from the set O such that there existed a clear ordering amongst them (e.g., for the

weight cylinders, two objects from the same pair would not be sampled together). The objects were then ordered (with the direction, forward or backward, determined randomly) and the last object was removed. Thus, the first three ordered objects formed the ordered set $\mathcal{L}$ for the given task. Three more objects were randomly sampled from the remaining objects in O such that none validly completed the ordering. These three objects, combined with the removed object, formed the set $\mathcal{G}$. The performance of the robot's model was evaluated in terms of accuracy, i.e., the number of tasks for which the robot's model picked the correct object divided by the total number of tasks.

For each concept, the performance of each sensorimotor context was evaluated. The accuracy was also computed as more and more contexts were used by the model. To estimate the context weights and to train the boosting method, five-fold cross-validation was performed with the 50 sampled tasks (i.e., 10 tasks were randomly assigned to each fold). The model was also evaluated as the number of tasks used for training was varied from 1 to 49. In this case, all contexts were used.

6. Results

6.1. An example order completion task

Fig. 3 shows an example task in which the robot's model is tasked with completing an order of three objects that are ordered by height. In this case, the ordered input set, $\mathcal{L}$, consists of three pink noodles, while the candidate set, $\mathcal{G}$, contains four objects – three cones and one noodle, such that only one of them is taller than the last element in $\mathcal{L}$. In this specific case, the input distance matrix encoded the perceptual similarity of the objects in the *press-proprioception* sensorimotor context. The figure shows an ISOMAP [25] embedding of the distance matrix, which makes it easy to see that the matrix encodes an order between the objects. For this task, the model correctly picked the cone from the set $\mathcal{G}$ that is taller than the tallest noodle object in $\mathcal{L}$. The next subsection describes a quantitative evaluation of the model in which each sensorimotor context is evaluated on each of the three ordering tasks.

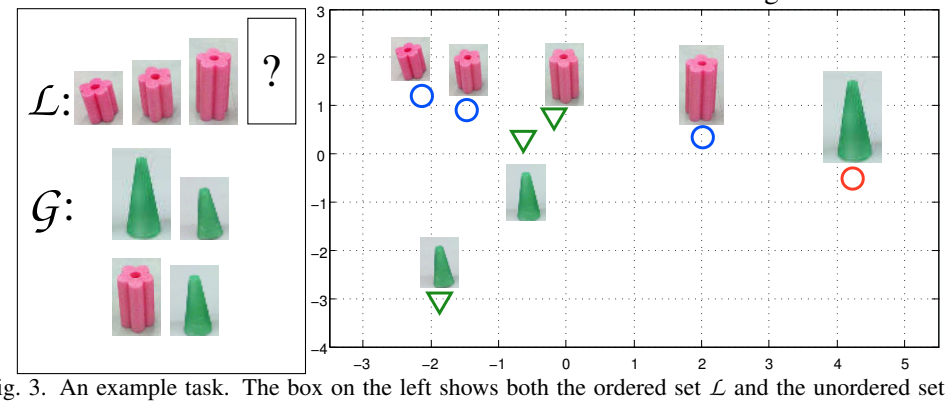


Fig. 3. An example task. The box on the left shows both the ordered set $\mathcal{L}$ and the unordered set of objects $\mathcal{G}$ to choose from. The plot on the right shows the ISOMAP embedding of the distance matrix between the objects. The blue circles denote the three objects in $\mathcal{L}$, the red circle denotes the object in $\mathcal{G}$ that is selected to complete the order.

6.2. Ordering objects using a single sensorimotor context

For the first experiments, the performance of the model was evaluated using a single sensorimotor context. Fig. 4 shows the accuracy for each context on each of the 3 concepts. As expected, *lift* (100%), *drop* (100%), *hold* (98.0%), *shake* (100%), and *rattle* (98.0%) for *proprioception* perform very well on the task of ordering objects by weight. This is likely because the robot was supporting the full weight of the object with its arm while performing these behaviors. For the *pressure cylinders*, *proprioception-lift* (100%) and *proprioception-tap* (98.0%) achieve high performance. The reason for this is likely due to the weight and moment of inertia differences in the objects caused by the different springs inside the *pressure cylinders*. *Proprioception-press* was able to achieve 100% accuracy on the *cones* and *noodles* task as was expected since the moment at which the arm touched the object varied depending on the object's height. The other sensorimotor contexts did not perform as well, with *proprioception-push* (84.0%) being the next highest performing context.

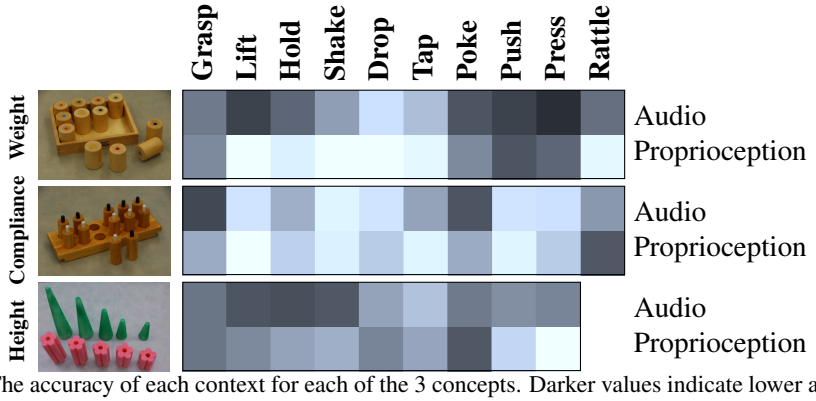
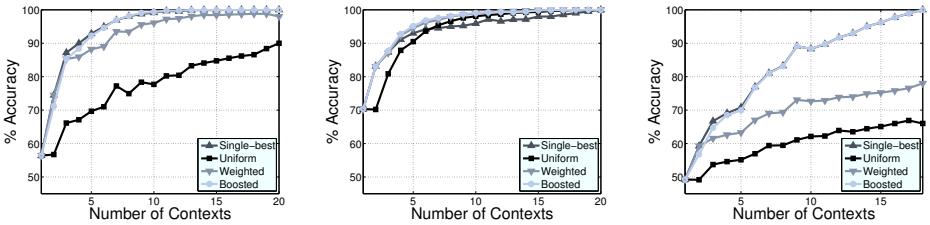


Fig. 4. The accuracy of each context for each of the 3 concepts. Darker values indicate lower accuracies with solid black being 0%; lighter values indicate higher accuracies with solid white being 100%.

6.3. Ordering objects using multiple sensorimotor contexts

Fig. 5 shows the performance of the robot on each concept as the number of contexts varies from 1 to $|\mathcal{C}|$ when using the uniform, weighted, and boosted combination methods as described in section 5.3.The accuracy when picking just the single-best context (based on the training tasks) is also shown for comparison. As the number of combined contexts increases, the average accuracy also increases, which is consistent with our previous results [17]. Additionally, in every case the weighted combination method outperforms the uniform combination method. Also the boosted method always does at least as well as the weighted method and in most cases outperforms it. For the *weight cylinders* (Fig. 5a), the weighted method reaches 98.0% accuracy and the boosted method reaches 100% accuracy when all contexts are used. For the *pressure cylinders* (Fig. 5b) the weighted and boosted combination methods reach 100% when all contexts are used. For the *cones* and *noodles* (Fig. 5c), the single-best context is able to achieve 100%, but when all the sensorimotor contexts are combined the robot was only able to achieve 72.0% accuracy using the weighted method. Using the boosted method, however, it was able to reach 100%. We believe that since for the

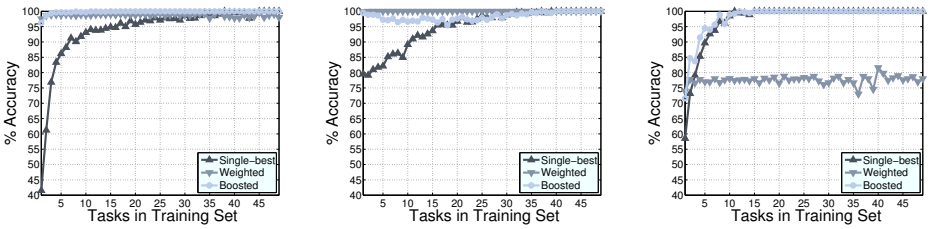


(a) Ordering by Weight (b) Ordering by Compliance (c) Ordering by Height

Fig. 5. The accuracy as the number of contexts is increased. The dark-gray line is the accuracy when picking the single-best context; the black line is the accuracy when using uniform weights to combine contexts; the gray line is the accuracy when the contexts are weighted in proportion to their individual accuracies; and the light-gray line is the accuracy achieved when using AdaBoost to learn the weights.

height concept, unlike for the other two, there was only one context that performed well, the noise from combining underperforming contexts outweighed the single best performing context for the weighted method, but the boosted method was able to learn this and weight the best context higher.

Fig. 6 shows the average accuracy as the number of tasks used for training is varied from 1 to 49 when combining all sensorimotor contexts. Again the single-best context (based on the training tasks) is shown for comparison. In every case, the weighted method converges after no more than 6 training tasks are used to estimate the weights. The boosted method always achieves 90% accuracy after no more than 4 training tasks and 95% accuracy after no more than 7. For *weight*, the boosted method and the weighted method converge at approximately the same rate. For *height*, the boosted method outperforms the weighted method by a large margin. For *compliance*, the boosted method converges slower than the weighted method (weighted reaches 100% after 3 tasks are used while boosted doesn't reach 100% until 40 tasks are used). This is likely related to the result in fig. 5, where *compliance* is the only task in which the uniform combination method reaches 100%. Interestingly, while the boosted method and the single-best context (as determined by the training set)



(a) Ordering by Weight (b) Ordering by Compliance (c) Ordering by Height

Fig. 6. The average accuracy as the number of tasks used for training is increased. The dark-gray line is the accuracy achieved when picking just the single-best context; the gray line is the accuracy achieved when the contexts are weighted in proportion to their individual accuracies; and the light-gray line is the accuracy achieved when using boosting. The results are averaged over 50 sets of training tasks for each size from 1 to 49.

converge to 100% for all three concepts, the boosted method converges much quicker for both the *weight cylinders* and *pressure cylinders*, and at about the same rate for the *cones* and *noodles*.

7. Conclusion and future work

In this paper we presented a theoretical model for performing order completion. We evaluated this model using an upper-torso humanoid robot on three concepts: *weight*, *compliance*, and *height*. The results show that the robot was able to select objects to complete orderings with a high degree of accuracy. For each concept, there existed at least one sensorimotor context that was able to achieve 100% accuracy, and there were multiple such contexts for *weight* and *compliance*. When combining sensorimotor contexts, on average, the best performance was achieved when all contexts were used, though in every case the best single context did at least as well or better. This suggests that when completing an ordering determined predominantly by only one property (e.g., weight), if there exists at least one sensorimotor context that is able to capture that property, then its predictions will typically align with the true ordering.

Given these results, what strategy should the robot use to solve a novel order completion task? The results clearly show that the boosted combination method is the best strategy for combining sensorimotor contexts because it always performs as well as or better than every other method and because it usually takes very few training tasks to train. The methodology used in this paper builds on our previous work, in which we have shown that stereotyped exploratory behaviors can be used to detect functional similarities between tools [18], perform object recognition [17], perform object categorization [20], recognize surface textures [21], solve the odd-one-out task [19], and now solve the order completion task. These results suggest that a wide variety of tasks can be solved using a library of task-specific algorithms applied on a common set of sensorimotor data extracted from exploratory behaviors.

A limitation of the method described in this paper is that while it can solve order completion tasks in which the order is ascending or descending by one property, it cannot solve more complicated tasks from the general domain of sequence completion. Therefore, future work will need to consider methods to solve completion tasks in which the transitions between elements are more complex than simply increasing or decreasing. Future work can also build on this model and others such as [19] and [9] that analyze the structure among groups of objects by using the discovered properties to scaffold learning of more complex concepts. Pursuing this line of research could allow robots to learn more complicated concepts that can be represented in terms of simpler concepts such as the ones explored in this paper.

References

1. Chapelle, O., Y. Chang, T. Liu. Future Directions in Learning to Rank. – In: JMLR: Workshop and Conference Proceedings, Vol. **14** , 91-100.
2. Ebeling, K., S. Gelman. Coordination of Size Standards By Young Children. – Child Development, Vol. **59**, No 4, 1988, 888-896.

3. Ebeling, K., S. Gelman. Children's Use of Context in Interpreting Big And Little. – Child Development, Vol. **65**, No 4, 1994, 1178-1192.
4. Forssén, P., D. Meger, K. Lai, S. Helmer, J. Little, D. Lowe. Informed Visual Search: Combining Attention And Object Recognition. – In: Proc. of IEEE International Conference on Robotics and Automation (ICRA), 935-942.
5. Freund, Y., R. Schapire. A Desicion-Theoretic Generalization of On-Line Learning And an Application to Boosting. – In: Computational learning theory, Springer, 23-37.
6. Graham, F., C. Ernhart, M. Craft, P. Berman. Learning of Relative And Absolute Size Concepts in Preschool Children. – Journal of Experimental Child Psychology, Vol. **1**, No 1, 1964, 26-36.
7. Hagmann-von Arx, P., C. Meyer, A. Grob. Assessing Intellectual Giftedness With The WISC-IV And The IDS. – Zeitschrift für Psychologie, Vol. **216**, No 3, 2008, 172-179.
8. Kaufman, A .. Intelligent Testing With the WISC-III. John Wiley & Sons, 1994.
9. Kemp, C., J. Tenenbaum. The Discovery of Structural Form. – Proceedings of the National Academy of Sciences, Vol. **105**, No 31, 2008, 10687-10692.
10. Lillard, A .. Montessori: The Science Behind the Genius. Oxford University Press, USA, 2008.
11. Lillard, A., N. Else-Quest. The Early Years: Evaluating Montessori. – Science, Vol. **313**, No 5795, 2006, 1893-1894.
12. Montessori, M .. The Montessori Method. Frederick A. Stokes Co., 1912.
13. Nakamura, T., T. Nagai, N. Iwahashi. Multimodal Object Categorization by a Robot. – In: Proceedings of the IEEE/RSJ IROS, 2415-2420.
14. Natale, L., G. Metta, G. Sandini. Learning Haptic Representation of Objects. – In: Proc. of the Intl. Conf. on Intelligent Manipulation and Grasping.
15. Nolfi, S., D. Marocco. Active Perception: A Sensorimotor Account of Object Categorization. – In: From Animals to Animats: 7, 266-271.
16. Pitamic, M .. Teach Me to Do It Myself. Elwin Street Productions, 2004.
17. Sinapov, J., T. Bergquist, C. Schenck, U. Ohiri, S. Griffith, A. Stoytchev. Interactive Object Recognition Using Proprioceptive And Auditory Feedback. – The Intl. J. of Robotics Research, Vol. **30**, No 10, 2011, 1250-1262.
18. Sinapov, J., A. Stoytchev. Detecting the Functional Similarities Between Tools Using a Hierarchical Representation of Outcomes. – In: Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL), 91-96.
19. Sinapov, J., A. Stoytchev. The Odd One Out Task: Toward an Intelligence Test for Robots. – In: Proceedings of the 9th IEEE International Conference on Development and Learning (ICDL), 126-131.
20. Sinapov, J., A. Stoytchev. Object Category Recognition by a Humanoid Robot Using Behavior-Grounded Relational Learning. – In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 184-190.
21. Sinapov, J., V. Sukhoy, R. Sahai, A. Stoytchev. Vibrotactile Recognition and Categorization of Surfaces by a Humanoid Robot. – Robotics, IEEE Transactions on, Vol. **27**, No 3, 2011, 488-497.
22. Sugarman, S .. The Cognitive Basis of Classification in Very Young Children: An Analysis of Object-Ordering Trends. – Child Development, Vol. **52**, No 4, 1981, 1172-1178.
23. Sun, J., J. Moore, A. Bobick, J. Rehg. Learning Visual Object Categories for Robot Affordance Prediction. – The Intl. J. of Robotics Research, Vol. **29**, No 2-3, 2010, 174.
24. Takamuku, S., K. Hosoda, M. Asada. Object Category Acquisition by Dynamic Touch. – Advanced Robotics, Vol. **22**, No 10, 2008, 1143-1154.
25. Tenenbaum, J., V. De Silva, J. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. – Science, Vol. **290**, No 5500, 2000, 2319-2323.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0018

Towards Autonomous Robotic Valve Turning

*A. Carrera *, S. R. Ahmadzadeh**, A. Ajoudani**, P. Kormushev**, M. Carreras*, D. G. Caldwell***

** Computer Vision and Robotics Group (VICOROB), University of Girona, 17071 Girona, Spain*
*** Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy*
Emails: ACarrera@eia.udg.edu {*Reza.Ahmadzadeh* |*Arash.Ajoudani* \ Petar.Kormushev}@iit.it *Marc.Carreras@udg.edu* *Darwin.Caldwell@iit.it*

Abstract: *In this paper an autonomous intervention robotic task to learn the skill of grasping and turning a valve is described. To resolve this challenge a set of different techniques are proposed, each one realizing a specific task and sending information to the others in a Hardware-In-Loop (HIL) simulation. To improve the estimation of the valve position, an Extended Kalman Filter is designed. Also to learn the trajectory to follow with the robotic arm, Imitation Learning approach is used. In addition, to perform safely the task a fuzzy system is developed which generates appropriate decisions. Although the achievement of this task will be used in an Autonomous Underwater Vehicle, for the first step this idea has been tested in a laboratory environment with an available robot and a sensor.*

Keywords: *Autonomous Underwater Vehicle (AUV), Imitation Learning, Fuzzy System, Extended Kalman Filter (EKF), Valve Turning.*

1. Introduction

Nowadays, Autonomous Underwater Vehicle (AUV) robots are used in different applications like seabed survey, mine cleaning, cable or pipeline tracking [1], deep ocean exploration with visual mapping [2] or water quality observation [3]. For

intervention tasks, still Remotely Operated underwater Vehicles (ROV) are used due to the complexity and uncertainty of the work. ROVs are operated by one or two persons, usually one to keep the robot stable and another to control the manipulators [4]. Recently, some positive results have been achieved in the task of recovering objects from the seabed ones using a robotic arm [5].

The main goal in the European project "Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning (PANDORA)"[6] is to develop and evaluate new computational methods to make human-built robots Persistently Autonomous. The goal is to reduce the frequency of assistance requests significantly and the key to this aim is the ability to recognize failure and respond to it autonomously. The PANDORA project is focused on three underwater tasks, one of them consisting in autonomous grasping and turning a valve with a free floating AUV.

The AUV uses a manipulator to grasp the correct valve on a panel and open or close it. Since the vehicle does not dock, it needs to hover by swimming when counteracting reaction forces from the turning and from the sea currents and even minor turbulence from the manifold. Also it must ensure that the gripper position and orientation of the gripper after grasping does not cause significant shear forces in the valve handle (T-bar shape), and break it off. To overcome the difficulties in this task, imitation learning techniques [7] can offer a robust solution and an easy way to teach the robot trajectory using the data from a set of demonstrations. This kind of a learning method includes the most important desirable properties of movement planning which are: ease of representing and learning, compactness of the representation, robustness against perturbations and changes in a dynamic environment, ease of reuse for related tasks and easy modification for new tasks, and ease of categorization for movement recognition. However, no standard approach of movement planning exists that accomplishes all these goals [8-11].

This paper presents the preliminary work about autonomous valve turning done with a real manipulator in lab conditions, not underwater. Several sensors have been used to estimate the distance between the gripper and the valve. Also, an Extended Kalman Filter (EKF) [12] has been applied to improve the estimations and to avoid gaps in the data. Moreover, according to the instant dynamics of the valve, the robot has to decide if it can approach to the valve or not. To make this decision, a fuzzy system has been used.

This paper has the following format. In Section 2 the general task and the proposed experimental environment are explained. In Section 3 the different methods and their combination to solve the task are described. The obtained results are showed in Section 4. And finally, conclusions are exposed in Section 5.

2. Problem description and environment

In PANDORA project, the task of grasping and turning the valve requires a set of actions that have to be done successfully before the final step. It must be noticed that in this paper the task of grasping and turning the valve is investigated, when the vehicle is in the work area in front of the desired valve.

This task will be accomplished by Girona500 [13], which is a compact and lightweight AUV, with hovering capabilities and can fulfil the particular needs of any application by means of mission-specific payloads and a reconfigurable propulsion system. In this case the AUV needs to be equipped with a robotic arm.

Before attempting the valve turning task underwater, an approach system has been built in the facilities of Istituto Italiano di Tecnologia (IIT), using a light weight robotic arm (KUKA/DLR) and an Optitrack system.

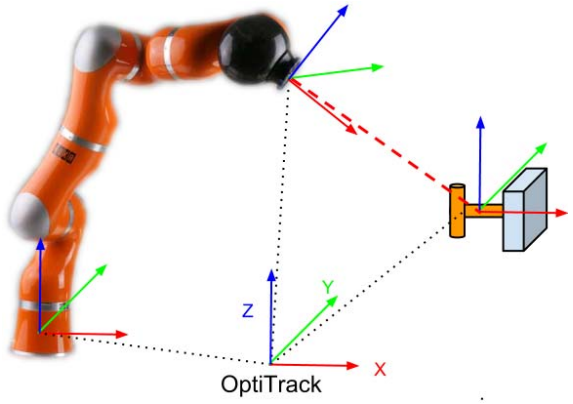


Fig. 1. Schematic diagram for the real scenarios

The KUKA/DLR robotic arm will be used under Cartesian impedance control mode [14]. The robot's position, orientation and the fixed joint or Cartesian stiffness commands will be sent to the KUKA controller using the DLR's Fast Research (FR) Interface libraries [15].

Although the KUKA/DLR is a different kind of manipulator than the one that is going to be attached to the Girona500 robot, the goal of this paper is to learn the attained trajectory during the experiment, not the specific kinematics of the manipulator. The available sensor in the AUV will be simulated using the Optitrack system, which lets you get the 3D position and orientation of a rigid body using a set of cameras and markers, see Fig. 1. This system gives the position and orientation with good precision and high frequency. In this problem we will focus our interest on the distance between the valve and the end effector; therefore we will mark these two elements to be tracked by the Optitrack system.

The real AUVs can acquire data from one camera, but the precision and sampling frequency of the Optitrack's data are much better. To improve the camera information, the robot has more sensors, in this case the gyro-enhanced Attitude and Heading Reference System (AHRS) will be used. The camera and the AHRS will be combined using an EKF to improve the estimation. Both sensors are simulated using the Optitrack's data.

Finally, the last difference between the two environments is the fact that underwater the valve will be fixed and the robot will move, but in the lab the situation is the opposite.

3. Experimental set-up and results

In this section the experimental set-up details, the process, and the results are described step by step. The diagram of the set-up is shown in Fig. 2.

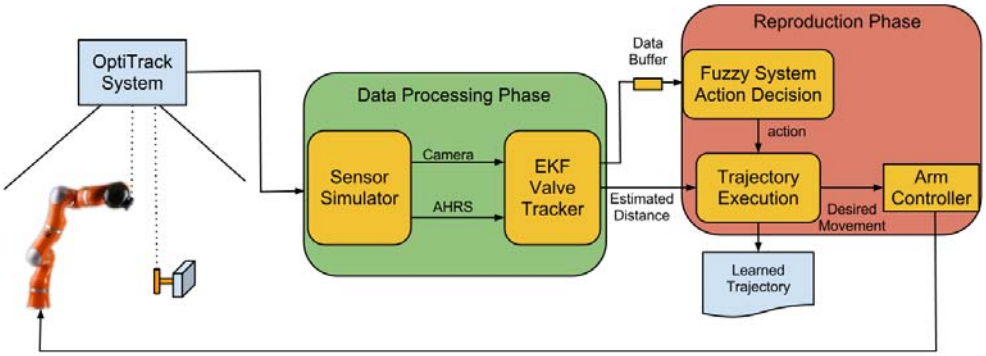


Fig. 2. Flowchart of the system: This diagram represents how the data flow from the Optitrack system to the different phases of experiment to finally convert in commands to the robot. The learning phase is not included because it is done offline with the recorded data

3.1. Data processing phase

In this phase, an EKF is designed to estimate the position. The EKF is an extension of Kalman Filter (KF) [12], able to work with non-linear functions. The KF uses measurements with noise and an approximated model of the process studied to produce measures of the state, which are more accurate than those based on single measurements. Previous works have proved the advantages of using an EKF to track objects [16, 17].

In this task the movement of the end effector towards the valve is represented as a model with a constant acceleration, and the measurements of the system are the position of the valve obtained using a camera, and the acceleration of the vehicle obtained using an AHRS. The orientation of the vehicle is also considered as a control signal.

The reference frame has the center in the valve position. In this way we avoid the differences between the two environments and solve the problem of changing the position of the valve. In this case, the mobile part in the coordinate system will be the end-effector and the base of the robot, all movements of the valve will be reflected in these two elements. The next equation shows the homogeneous transformation Matrix done to convert the data from the Optitrack frame to the valve frame, in this case a composed matrix is used with translation and standard rotation with RPY Matrix:

$$ {}^{\text{optitrack}}K_{\text{valve}} = \begin{pmatrix} & & Px_{\text{valve}} \\ R(\alpha_{\text{valve}}, \beta_{\text{valve}}, \gamma_{\text{valve}}) & & Py_{\text{valve}} \\ & & Pz_{\text{valve}} \\ 0 \quad 0 \quad 0 & & 1 \end{pmatrix}. \tag{1} $$

The two inputs of EKF are generated using different samples rates from the Optitrack's data. The data selected to generate one sensor will never be used to generate the data of the other, making both sensors independents. Moreover, appropriate noise is added to the signals, using a precision similar to the sensors, for the camera it is ± 0.005 m, and for the AHRS it is ± 0.5 m/s^2.

3.2. Learning phase and trajectory execution

A robot should be able to acquire new skills using various forms of learning and when direct physical contact to the robot is possible, kinaesthetic teaching offers a user-friendly and intuitive method to demonstrate new skills to a robot by manually guiding the robot's arm through the motion.

Briefly, using several demonstrations of a similar task, the robot creates a compact model of the skill by taking into account the variations and correlations observed along the movement. The positional constraints of the demonstrated skill are represented as a matrix of dynamical systems encoding robustly position trajectories. The Dynamic Movement Primitives (DMP) framework, which is used in this paper, was originally proposed by I j s p e e r t et al. [7] and after that in [18-20].

M examples of a skill are demonstrated to the robot with different initial positions. Each demonstration $m \in \{1,\ldots,M\}$ consists of a set of T_m positions x, velocities $\dot{x}$ and accelerations $\ddot{x}$ of the end-effector in Cartesian space, where each position has three dimensions. Using the datasets from demonstrations, a mixture of K proportional-derivative systems is created as a model of the skill [20].

In this approach a decay term defined by a canonical system $\dot{s} = -\alpha s$ is used to create an implicit time dependency property $t = -\ln(s)/\alpha$, in which the initial value for s is 1 and converges to zero. Also for the backward movement, which is used in retracting mode, a complementary equation is used to generate time starting from the final time to the initial time.

A set of *K* Gaussians is defined in time space, with centres μ_i^{T} equally distributed in time, and variance parameters set to a constant value inversely proportional to the number of states. By determining the weights $h_i(t)$ through the decay term s, the system will sequentially converge to the set of attractors in Cartesian space defined by centres μ_i^X and stiffness matrices k_i^P, which are learned from the observed data, either incrementally or in a batch mode.

The desired acceleration to generate the trajectory is computed using the next equation, where x and $\dot{x}$ are the current position and velocity and k^v defines the damping factor:

$$\hat{\ddot{x}} = \sum_{i=1}^{K} h_i(t)\left[k_i^P \left(\mu_i^X - x \right) - k^v \dot{x} \right]. \quad (2)$$

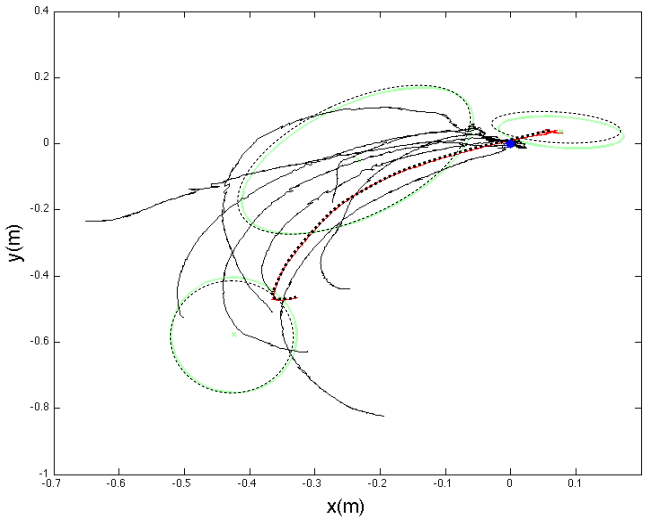


Fig. 3. In dense line you can see all the trajectories in 2D, given by demonstration. The broken line ellipsoids are the states learned by the algorithm to represent the trajectory and in dotted line you can see one trajectory produced using the learning part

It can be seen that for parts of the movement where the variations across the different demonstrations are large, the reference trajectory does not need to be tracked precisely. By using this information, the controller can focus on the other constraints of the task, such as collision avoidance. On the other hand, for other parts of the movement, exhibiting strong invariance across the demonstrations should be tracked precisely.

3.3. Reproduction phase

A fuzzy system is used to generate a decision command based on linguistic variables and rules. In a fuzzy system the fuzzifier section maps the crisp inputs into some fuzzy sets. Then the fuzzy inference engine uses fuzzy IF-THEN rules from the defined rule base to reason for the fuzzy output. The generated output in a fuzzy term is converted back to the crisp value by the defuzzifier section [21].

Since we are using EKF to make estimation and fill in the gaps from our sensor data, when we do not receive data for a while the uncertainty becomes bigger and bigger. In addition, the robot or the operator needs to know about the dynamics of the valve and decide if it can be approached, because if the relative movement exceeds the normal range we may miss the valve or break it off. Therefore, the first

input for our fuzzy system is an estimated movement between the valve and the arm. And the second input is receiving the data delay from the sensor. The output of the fuzzy system is a numerical command in the continuous range of grasping, waiting, and retracting actions [–1, 1].

Here, we use Sugeno inference which consists of product inference engine, singleton fuzzifier, and center average defuzzifier. We use Gaussian membership functions in our fuzzy sets, and the designed fuzzy rule base is showed in Table 1.

Table 1. Fuzzy rule base

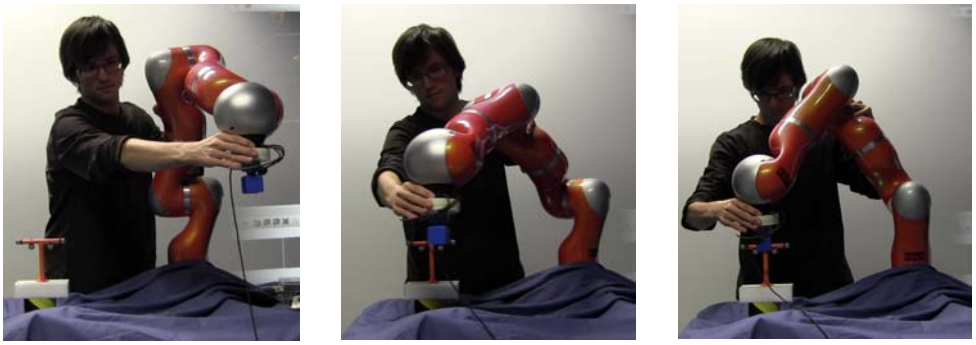
		Relative Movement		
		Small	Medium	Big
Sensor Delay	Low	Forward	Stop	Backward
	Medium	Forward	Stop	Backward
	High	Stop	Backward	Backward

After defining the rules for the system, let w be the output value of each step and z be the weight for each rule, then the final output of the fuzzy system is [21]:

$$\text{FinalOutput} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}. \tag{3}$$

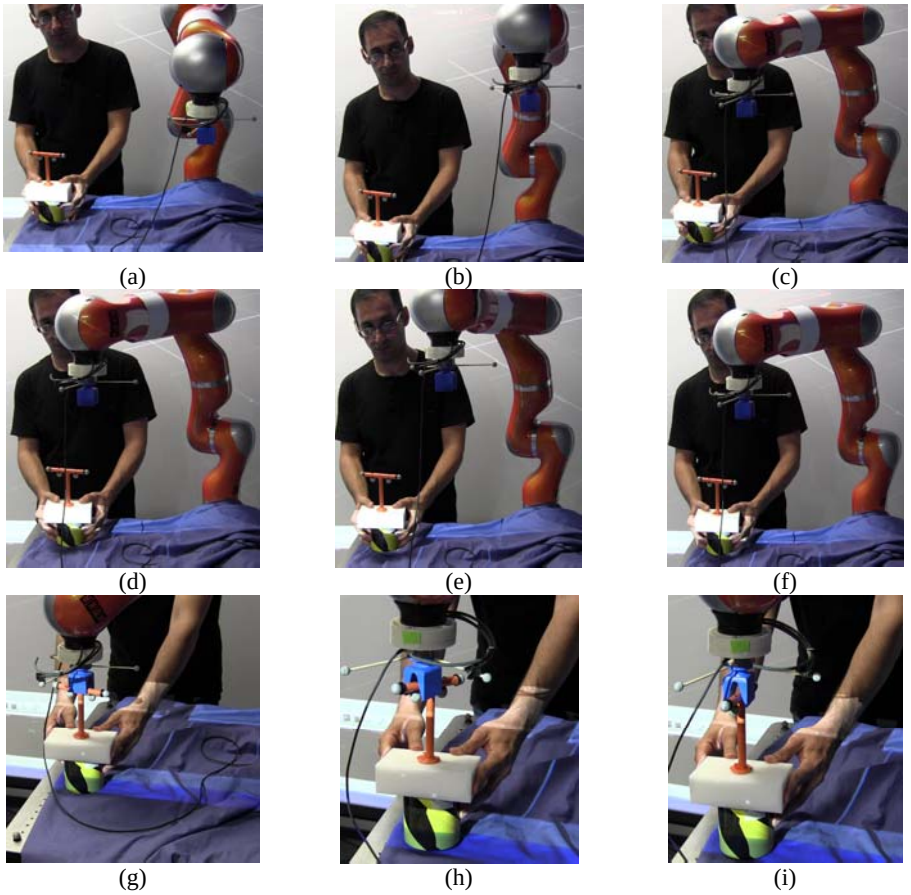
Finally, with the input of the distance between the end effector and the valve, and with the decision of the fuzzy system the learning part generates a proportional movement with respect to the position to move the robot. The result is evaluated and if the condition of grasping the valve is completed, an instruction is sent to turn the valve, in the other case the new position is sent to the KUKA robotic arm.

3.4. Complete experiment



(a) (b) (c)

Fig. 4. This set of images represents the process of one demonstration of the task of grasping the valve



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 5. The set of images show the whole process of reproducing the task with perturbations. Images (a) and (b), the robot reaches the initial common position, from any position. Images (c) and (d), the valve is not stable and the robot moves to a safer position. Image (e). Image (f), the valve is stable in a new position, so the robot moves to grasp it. Image (g), the robot has grasped the valve and finished the trajectory. Images (h) and (i), the robot does the 90º turning

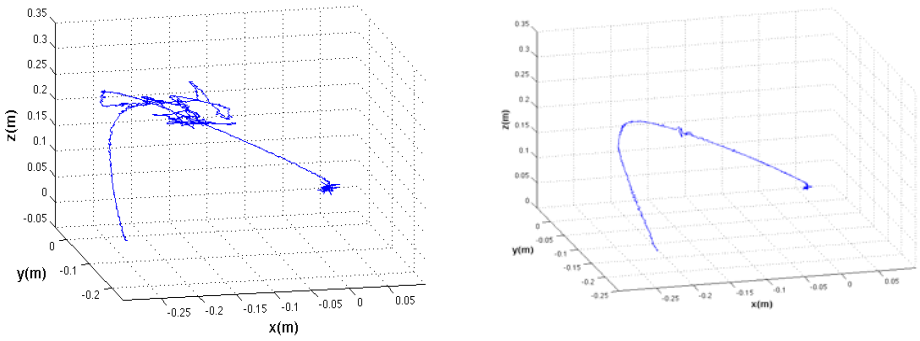


Fig. 6. In this set of images we can see two different trajectories which are done by the end effector to grasp and turn the valve. In the left figure we can see the changes in the trajectory, moving forward or backward, depending on the stability of the valve position. On the other side, the trajectory on the right figure shows a clean trajectory going only forward to the valve, in this case the valve has been stable during the process

In the final experiment all steps of the operation (from learning until valve turning) in a complete loop are accomplished by the robot appropriately and effectively, see Figs 4, 5 and 6. The designed system is capable of providing a smooth trajectory, compensating small perturbations in sensors, analysing the safeties of the situation and moving forward or backward with smooth changes, tracking the trajectory efficiently, and detecting the end of the trajectory and turning the valve. In future works the turning phase will be learned by the robot instead of using pre-programmed commands.

4. Conclusions

The proposed combination of techniques in this paper has allowed a robotic arm to learn the skill to follow a trajectory, grasp a valve and turn it. This experiment has been done as a simulated scenario of an underwater operation with an AUV.

During this experiment the robot has been tracking the distance between the valve and the end effector of the robotic arm, doing a mixture of different kind of measurements using the EKF which generates a smooth movement of the robot and more stability in the control of the position. Moreover, the learning part has provided the ability to extract the important restrictions of a set of trajectories and offer the adaptability and robustness to follow the trajectory. In addition, the method has proved the possibility of utilizing a fuzzy system to study the dynamic behaviour of the valve and choose a proper action.

Acknowledgement. This research was sponsored by PANDORA[6] EU FP7-Project under Grant agreement No ICT-288273.

References

1. Gracias, X., S. Negahdaripour. Underwater Mosaic Creation Using Video Sequences from Different Altitudes. – In: OCEANS, 2005. Proceedings of MTS/IEEE, Vol. **2**, 17-23 September 2005, 1295-1300.
2. Billar, S. Calinon, R. Dillmann, S. Schaal. Survey, Robot Programming by Demonstration. Handbook of Robotic. Chapter 59. 2008.
3. Guo, S., X. Lin. Development of a Vectored Water-Jet-Based Spherical Underwater Vehicle. – In: N. A. Cruz, Ed. Autonomous Underwater Vehicles. InTech, Janeza Tardin 9, 51000 Rijeka, Croatia, 2011, 3-20.
4. Ryu, J. H. Control of Underwater Manipulators Mounted on an ROV Using Base Force Information. – In: IEEE Int. Conf. on Robotics and Automation (ICRA), Vol. **4**, 2001, 3238-3243.
5. Prats, M., D. Ribas, N. Palomeras, J. C. García, V. Nannen, S. Wirth, J. J. Frenández, J. P. Beltráns, R. Campos, P. Ridao. Reconfigurable AUV for Intervention Missions: A Case Study on Underwater Object Recovery. – Intelligent Service Robotics, Vol. **5**, 2012, 19-31.
6. Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning (PANDORA). **http://persistentautonomy.com/**
7. Ijspeert, A. J., J. Nakanishi, S. Schaal. Trajectory Formation for Imitation with Nonlinear Dynamical Systems – In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), Maui, USA, 2001, 752-757.
8. An, C. H., C. G. Atkeson, J. M. Hollerback. Model-Based Control of a Robot Manipulator. MIT Press, 1988.

9. Miyamoto, H., S. Schaal, F. Gandolfo, Y. Koike, R. Osu, E. Nakano, Y. Wada, M. Kawato. A Kendama Learning Robot Based on Bi-Directional Theory. – Neural Networks, Vol. **9**, 1996, 1281-1302.
10. Kawato, M. Trajectory Formation in Arm Movements: Minimization Principles and Procedures. – In: H. Z. Zelaznik, Ed. Advanced in Motor Learning and Control. Champaign Illinois, Human Kinetics Publisher, 1996, 225-259.
11. Sutton, R., A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.
12. Welch, G., G. Bishop. An Introduction to the Kalman Filter. ACM SIGGRAPH 2001, Course 8.
http://ww.cs.unc.edu/~welch/kalman/
13. Ribas, D., N. Palomeras, P. Ridao, M. Carreras, A. Mallios. Girona 500 AUV, from Survey to Intervention – IEEE/ASME Transactions on Mechatronics, Vol. **17**, 2012, 46-57.
14. Albu-Schäffer, A., C. Ott, G. Hirzinger. A Unified Passivity-Based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. – Int. Journal of Robotics Research, Vol. **26**, 2007, No 1, 23-39.
15. Schreiber, G., A. Stemmer, R. Bischoff. The Fast Research Interface for the KUKA Lightweight Robot. – In: IEEE ICRA, 2010.
16. Funk, N. A Study of the Kalman Filter Applied to Visual Tracking. Technical Report. University of Alberta, 2003.
17. Zhou, Q., J. K. Aggarwal. Object Tracking in an Outdoor Environment Using Fusion of Features and Cameras. – Image and Vision Computing, Vol. **24**, 2006, 1244-1255.
18. Schaal, S., P. Mohajerian, A. J. Ijspeert. Dynamics Systems Vs. Optimal Control a Unifying View. – Progress in Brain Research, Vol. **165**, 2007, 425-445.
19. Hoffmann, H., P. Pastor, D. H. Park, S. Schaal. Biologically-Inspired Dynamical Systems for Movement Generation: Automatic Real-Time Goal Adaptation and Obstacle Avoidance. – In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), Kobe, Japan, May 2009, 2587-2592.
20. Kormushev, P., S. Calinon, D. G. Caldwell. Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input. – Advanced Robotics, Vol. **25**, 2011, No 5, 581-603.
21. Wang, L. X. A Course in Fuzzy System and Control. 2nd Edition. Prentice-Hall International, Inc., 1997. ISBN: 0135408822.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0019

Learning to Generalize from Demonstrations

Katie Browne, Monica Nicolescu

University of Nevada, Reno, NV 89557, USA
Emails: katiembrowne@gmail.com monica@cse.unr.edu

Abstract: *Learning by demonstration is a natural approach that can be used to build a robot's task repertoire. In this paper we propose an algorithm that enables a learner to generalize a task representation from a small number of demonstrations of the same task. The algorithm can generalize a wide range of situations that typically occur in daily tasks. The paper also describes the supporting representation that we use in order to encode the generalized representation. The approach is validated with experimental results on a broad range of generalizations.*

Keywords: *Learning by Demonstration, Generalized Representation, Graph Task Representation, Behavior Graphs, Robotics.*

1. Introduction

The ability to learn new knowledge is essential for any robot to be successful in real-world applications where it would be impractical for a robot designer to endow it with all the necessary task capabilities that it would need during its operational lifetime. Therefore, it becomes necessary that the robot is able to acquire this information in a human-like teaching approach. While significant research has been performed in the area of learning motor behavior primitives from a teacher's demonstration, the topic of learning of the general task knowledge has not been sufficiently addressed. In learning general task knowledge, the main challenge for

the learner is to extract all the necessary information pertaining to the task, eliminate all the observations that are irrelevant and generalize the correct task representation in the case when multiple, possibly different demonstrations are given. In this paper we propose an algorithm that enables generalization of correct and complete task knowledge from a small number of demonstrations of the same task, under the assumption of possibly very different demonstrations provided to the learner.

In our proposed approach we will assume that a learner robot is equipped with a basic set of capabilities or skills; the robot is aware of the goals each of these skills accomplish and also under which conditions these skills can or should be executed. During the learning stage, the learner robot is presented with a set of training examples with which it makes a generalized representation graph structure for each of the skills. These generalized representation graphs are then used in the execution part to create an execution sequence that achieves the task at hand.

A significant advantage of the proposed method is these generalized representation graphs are always consistent with the training samples. In addition, our approach allows the robot to execute the learned tasks in collaboration with a human user: the user can help the robot by performing some parts of the task and the robot takes into account the user's actions in order to finish up the task.

The remainder of the paper is structured as follows: Section 2 summarizes related work in the field of learning by demonstration. Section 3 presents the generalization problems this paper is attempting to address and it describes the proposed method used to solve these problems. Section 4 presents the experimental results, Section 5 discusses these results and future work, and finally Section 6 concludes our paper.

2. Related work

A significant challenge in designing robot systems that learn from demonstration is the interpretation of observations gathered from the instruction, as the robot has to process the continuous stream of data coming from its sensors, and then translate it into appropriate skills or tasks. In most cases this consists of segmenting the data stream into meaningful units, and then mapping them to a set of existing behavior primitives (Schaal [1]). There is a large spectrum of approaches to the problem of segmentation, including Principal Component Analysis (PCA) (Voyles, Morrow & Khosla [2]), gesture interpretation (Voyles & Khosla [3]), Learning Vector Quantization (LVQ) (Pook & Ballard [4]), motion contact (Tominaga, Takamatsu, Ogawara, Kimura&Ikeuchi [5], Ikeuchi, Kawade & Suehiro [6]) and geometric interpretation of the demonstration (Kuniyoshi&Inoue[7]).

Several recent approaches to learning by demonstration also use a graph/tree representation. Feature decision trees have been used to find the next behavior for a soccer player (Avrahami-Zilberbrand & Kaminka [8]), topological task graphs using longest common sequences has been used in path planning

(Abbas & MacDonald [9]), and skill trees have been successfully applied to the pinball domain (Konidaris, Kuindersma, Grupen & Barto [10]). All of these approaches differ from the generalization graphs proposed in this paper in that the graph/trees represent different elements of the learned tasks.

3. Task learning from generalizations

3.1. Classes of generalization problems

Approaches to learning by demonstration or from observation are significantly influenced by performance of the demonstrator. The challenge for the learner is to extract all the relevant information pertaining to the task and eliminate all observations that are irrelevant. During learning from demonstrations there are various aspects of the task that could and should be refined through generalizations. In this paper our goal is to address the following generalization problems:

1. *Generalization 1*. Learning to distinguish between:
    a. Tasks for which the ordering of the demonstration steps is important. Under this case, the exact demonstrated path needs to be reproduced.
    b. Tasks for which the ordering of the task's steps is not important, but where achieving the final goal is the main focus of the demonstration. Under this case, multiple ways of achieving the goal could be possible.
    c. Tasks for which the ordering of certain steps is important but the ordering of the rest of the steps is not important. Under this case,
        i. A step or set of steps needs to come before another step or set of steps but is unrelated to the other steps in the task.
        ii. A sequence of steps may need to be performed together but can appear anywhere during the task.
        iii. A step may need to be done at the same point in time.
2. *Generalization 2*. Learning to distinguish which steps in the demonstration are relevant, and which steps do not bring any contribution to the task. This will be done by monitoring the steps that do not appear in every demonstration.
3. *Generalization 3.* Learning to distinguish the number of times a step needs to be repeated since this number may change in each demonstration. This will be done by monitoring the post conditions of the repeated step after all the repetitions.

3.2. Collaborative behavior during task execution

Another goal of the proposed approach is to handle situations, in which a teacher or another user helps out the robot that is currently carrying out a learned task. In this case the learner will need to detect that some of the behaviors may have been done while the learner was performing a different behavior in the task. The learner also needs to make sure the behavior that the other person performed met the post conditions of the behavior and if not, the learner will need to complete that behavior.

3.3. Representation of generalized tasks

In this work a task is represented as a sequence of symbols where each symbol corresponds to a behavior that a robot can perform. A sequence of such symbols, provided from a teacher's demonstration, constitutes a training sample. In this context, a learner is given a set of sequences, potentially different, that all achieve the same target task. We base our generalization strategy on the preconditions of behaviors shown in the demonstration, based on what other behaviors have occurred prior to their execution. For instance, if behaviors *a* and *b* come before *d* in all training samples, but *c* can appear before or after *d*, we can generalize that behaviors *a* and *b* are preconditions for behavior*d* as they may achieve goals that are necessary for *d*'s execution. Our goal is to build, for each behavior present in the demonstrated task, a generalized representation of behavior preconditions that encapsulates the information contained in all training samples. This representation could afterwards be used by the learner to perform the same observed task.

The representation for generalized preconditions in this approach is a graph that is mainly a tree structure with the added feature that leaf nodes in the graph can contain an edge pointing back to the root of the graph. The root of the graph is a behavior for which we build the generalized preconditions and the branches of the graph are the behaviors that came before the current behavior in the training examples. Thus, each behavior in the training example has its own graph.

To demonstrate how the generalized precondition graphs for a task are created, let us consider an example in which the following training samples have been given for a particular task:

- *abcd*
- *cadb*
- *cdba*

Our approach works by starting at the first behavior in the first sequence which in this case is *a*. A new graph is created with *a* becoming the root node and a node with the name *NULL* added as a child, because no behavior has been executed before *a*, and therefore the behavior has no preconditions. This child node is then marked as final state; the final state signifies that the behavior represented by the root node can be performed after the behavior marked as a final node in the current sequence. Also the number of times a final node appears before the behavior is recorded and is used at the execution stage to decide which behavior to perform in a sequence when multiple behaviors are applicable and can be chosen at the same time. Next the algorithm creates the precondition graph for behavior *b*; the graph starts with *b* as the root node and *a* as a final state child node, because *a* was executed before it in the first training sample. A new graph is then created for behavior *c* and the branch with nodes *a* and *b* is added. Finally the graph for *d* is created and *a*, *b*, and *c* are added as a branch. The graphs created are shown in Fig. 1. The numbers appended to the behaviors in the graph have the following meaning: the first number appended signifies the level that it is located in the graph and the second number is only appended to the final state nodes, representing the number of times the node was a final state in the training samples.

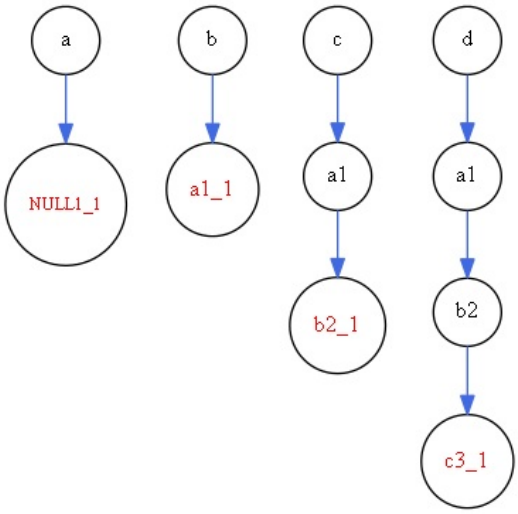


Fig. 1. Graphs created after the first sequence of behaviors

After the second training sequence is added, all the precondition graphs are refined as follows. Behavior*c* is the first in the training sample, so a null child node is added to its graph, indicating that it can also be the first one executed in the task. Then, behavior *c* is added as a child node in behavior *a*'s graph, a branch containing behaviors*c* and *a* are added to *d*'s graph, and finally *c*, *a*, and *d* become a branch in *b*'s graph. Fig. 2 shows the graphs after adding the second training sample.

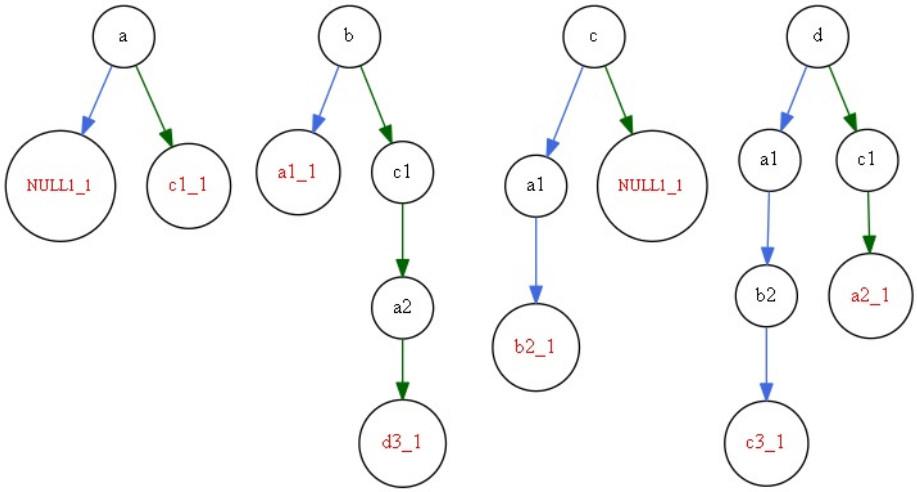


Fig. 2. The behaviour graphs after the second sequence has been added

Lastly the third training sample is incorporated using the same strategy as above. The resulting generalized precondition graphs can be seen in Fig. 3.

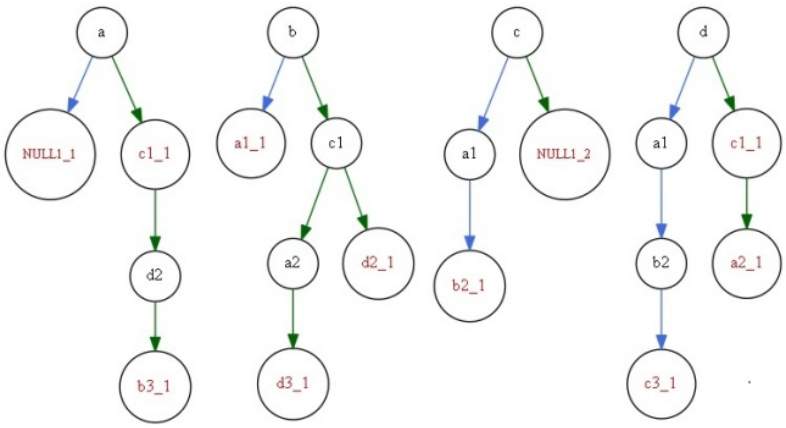


Fig. 3. The behaviour graphs after the third and last sequence has been added

If a behavior occurs in a training sample two or more times, the children nodes in the generalized precondition graph will point back to the parent node. Consider the sequence:

- *abcfdefg*

When a graph for behavior *f* is created, a branch containing the behaviors *a*, *b*, and *c* will be added to *f*'s graph and the *c* node will have an edge pointing to *f*'s node. A branch containing *d* and *e* is also added to the graph. *f*'s graph can be seen in Fig. 4.

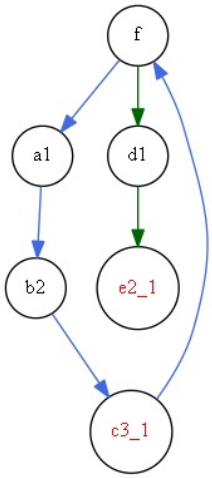


Fig. 4. Example of a graph whose behaviour appears one or more times in a sequence

Additionally, the graphs keep track of the number of training samples each behavior appears in. This number can be used to calculate the percentage of times a behavior appeared in the training example which in turn is used to find behaviors that did not appear in every sample. Depending on this percentage, we can choose to keep the behavior's graph or completely remove it.

Finally, the generalized precondition graphs keep track of whether a behavior appeared two or more times in a training sample. In our representation, these behaviors are aggregated into *special behaviors* which are just behaviors that the

learner repeats in the task as efficiently as possible until the post conditions are met. At the beginning of the learning period, the learner is provided a list of behaviors that can be performed and the possible parameters for these behaviors. Conclusively, these special generalized representation graphs are always consistent with the training samples.

3.4. Usage of generalization graphs

The generalized representations we built through the method presented above can be used by the learner to reproduce the task that has been demonstrated. Given that multiple demonstrations have been provided, the learner needs to find an execution sequence that achieves the same goal and is consistent with all the training examples. To do this, the system goes through all the precondition graphs and finds the behaviors that have the current execution sequence as a branch in their graph. If two or more behaviors fit the criteria, we pick the behavior that has been visited the most at the current time behavior in the training examples or if all the behaviors have been visited equally, we randomly pick a behavior as the next one in the sequence. To demonstrate this, considered the previous example and graphs in Fig. 3:

- t_0: Current sequence: *null* Behaviors that qualify: *a* and *c* Choose: *c*
- t_1: Current sequence: *c* Behaviors that qualify: *a* and *d* Choose: *d*
- t_2: Current sequence: *cd* Behaviors that qualify: *b* Choose: *b*
- t_3: Current sequence: *cdb* Behaviors that qualify: *a* Choose: *a*
- t_4: Current sequence: *cdba*

4. Experimental results

The system was tested in the context of a task for making the dough for chocolate chip cookies. A training example was a set of sequences that were made up of strings representative for the behaviors needed for this task. The list of behaviors for this task and the parameters that these behaviors can take are as follows:

Add Sugar (*Su*) 0.5 cup,
Add Salt (*Sa*) 1 tsp,
Add Baking Soda (*BSo*) 1 tsp,
Add Flour (*F*) 0.5, 1, 2 cup,
Add Chocolate Chips (*CC*) 0.5, 1, 2 cup,
Add Brown Sugar (*BSu*) 1, 2 cup,
Add Vanilla (*V*) 1 tsp,
Add Butter (*B*) 0.5, 1, 2 cup,
Add Eggs (*E*) 1, 2 egg,
Add Nuts (*N*) 0.5, 1, 2 cup,
Stir (*St*) 1, 2, 3, 4 time.

The results for our system depended on whether or not the generalized representations produced an execution trace consistent with all the training examples. In most cases, this meant that the order of the behaviors matched one of the sequences given to our system. However, when the examples had repeated

behaviors or behaviors that only appeared in a small percent of the sequences, this was not necessarily the case. When there were repeated behaviors, the number of times the behavior was repeated depended on what amounts the learner could add to meet the post conditions. Furthermore, when a behavior appeared in a small percent of the input sequences, the behavior did not appear in the execution sequence when it was in less than 50% of the input sequences. In addition, as mentioned earlier, in the execution phase the robot can collaborate with a human user, who may help the robot by performing some of the steps in its task. This human intervention may alter the path that the learner would have chosen to perform the task, while still obeying all the constraints of the training samples. Also, if the human user performs a behavior only partially the learner will need to complete it. When a human user performs an incomplete behavior, the number of times a behavior was repeated may not match a seen training example but it will obey the rules represented in the training examples. It is important to note that the human user can perform any behavior in the execution sequence but the following examples show only the cases where user input had a significant impact on the final execution sequence. In all of the examples the numbers in the parenthesis represent the parameter values for each behavior; the units can be found in the list above. We tested eight different scenarios, which cover all of the generalization cases presented in Section 3. These examples and the results are presented below.

Scenario 1. This constitutes an example where the training sequences were identical and it shows that the system can correctly encode tasks from generalization 1.a:

- *F*(2)*Sa*(1) *BSo*(1)*B*(2)*BSu*(1)*Su*(0.5)*V*(1) *E*(2) *CC*(2)
- *F*(2)*Sa*(1) *BSo*(1)*B*(2) *BSu*(1)*Su*(0.5)*V*(1) *E*(2) *CC*(2)
- *F*(2) *Sa*(1)*BSo*(1)*B*(2) *BSu*(1)*Su*(0.5)*V*(1) *E*(2) *CC*(2)

There is no variation in the training examples and thus the execution sequence produced by the system is identical to the training examples:

- *F*(2)*Sa*(1) *BSo*(1) *B*(2) *BSu*(1) *Su*(0.5) *V*(1) *E*(2) *CC*(2)

In this scenario, the helper performs the behavior add one egg which does not fully achieve the postconditions of that behavior.Our program is able to perform the behavior again to create an execution sequence consistent with the training examples; the execution sequence looks as follows:

- *F*(2)*Sa*(1) *BSo*(1)*B*(2) *BSu*(1) *Su*(0.5) *V*(1) *E*(1) *E*(1) *CC*(2)

Scenario 2. This scenario aims to illustrate that the system can identify and encode the type of generalizations described in 1.b:

Example 1. In a first setup, the learner is provided with the following demonstrations:

- *F*(2) *Sa*(1) *BSo*(1) *B*(2) *BSu*(1) *Su*(0.5)*V*(1) *E*(2) *CC*(2)
- *CC*(2)*V*(1) *F*(2) *BSu*(1) *E*(2) *Su*(0.5) *B*(2) *Sa*(1) *BSo*(1)
- *BSo*(1) *E*(2)*BSu*(1) *V*(1) *Su*(0.5)*Sa*(1)*CC*(2) *B*(2)*F*(2)

In this scenario the training examples were all different, with very few ordering constraints that are consistent throughout all the examples. The system

produces an execution sequence similar to one of the training examples (in this case, the third):

- *BSo*(1) *E*(2)*BSu*(1) *V*(1) *Su*(0.5) *Sa*(1) *CC*(2) *B*(2) *F*(2)

If in this example a helper performs a behavior at the beginning of the task, then the robot may take a different path to finish the task. The execution sequence above was chosen at random but if the helper performs the behavior *F* first, then the following sequence will be executed:

- *F*(2) *Sa*(1) *BSo*(1)*B*(2) *BSu*(1)*Su*(0.5)*V*(1)*E*(2) *CC*(2)

Example 2. In a first setup, the learner is provided with the following demonstrations:

- *BSo*(1)*Sa*(1) *Su*(0.5) *CC*(2) *E*(2)*F*(2) *BSu*(1)*B*(2) *V*(1)
- *V*(1) *F*(2) *B*(2) *Su*(0.5) *E*(2) *CC*(2) *Sa*(1) *BSu*(1) *BSo*(1)
- *B*(2)*E*(2) *CC*(2) *Sa*(1) *BSo*(1)*BSu*(1)*F*(2)*V*(1)*Su*(0.5)
- *E*(2)*F*(2) *Su*(0.5)*BSu*(1)*CC*(2) *V*(1)*Sa*(1) *B*(2)*BSo*(1)
- *B*(2)*E*(2)*CC*(2)*BSo*(1) *Sa*(1) *Su*(0.5) *F*(2)*V*(1) *BSu*(1)

In this scenario the training examples were all different, with no ordering constraints that are consistent throughout all the examples. There were two cases that started with the same three steps and thus the robot choose to start with those three steps since they had occurred the most in those positions. At this point in the program, the approach will randomly choose between *Sa* and *BSo*. In this example the helper assisted the robot after the first three steps, by performing *BSo*. Our approach was able to successfully find the next step that obeyed all the rules of the training data.

- *B*(2)*E*(2) *CC*(2)*BSo*(1) *Sa*(1) *Su*(0.5) *F*(2)*V*(1) *BSu*(1)

Scenario 3. This scenario falls under the generalization problem 1.c.i:

- *CC*(2)*Sa*(1) *V*(1) *Su*(0.5)*BSu*(1) *B*(2)*BSo*(1)*E*(2)*F*(2)
- *BSo*(1) *F*(2)*BSu*(1) *Sa*(1)*CC*(2)*E*(2) *B*(2)*Su*(0.5) *V*(1)
- *Su*(0.5) *V*(1)*B*(2) *CC*(2) *Sa*(1)*F*(2)*BSu*(1) *E*(2)*BSo*(1)

The above scenario is representative of cases in which a behavior needs to be executed before another behavior but not necessarily immediately before. In the training sequences above, add salt (*Sa*) always came before add eggs (*E*) but never right before it. The system produces an execution sequence similar to one of the training examples (in this case, the first):

- *CC*(2)*Sa*(1) *V*(1) *Su*(0.5) *BSu*(1)*B*(2)*BSo*(1)*E*(2)*F*(2)

Scenario 4. This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.c.ii:

- *CC*(2)*F*(2) *Sa*(1) *BSo*(1) *BSu*(1) *B*(2) *E*(2) *V*(1) *Su*(0.5)
- *F*(2)*Sa*(1)*BSo*(1)*BSu*(1) *Su*(0.5)*CC*(2)*B*(2)*E*(2)*V*(1)
- *B*(2)*E*(2) *V*(1) *Su*(0.5) *CC*(2) *F*(2)*Sa*(1) *BSo*(1)*BSu*(1)

In this set of training examples, *F* always came before *Sa* which always came before *BSo* and *BSo* always came before *BSu*. Also *B* always came before *E* which always came before *V*. Fig. 5 shows the precondition graphs for two of the behaviors, one is a generalized behavior, *F+Sa+BSo+BSu*, and *CC*. The system

produces an execution sequence similar to one of the training examples (in this case, the third):

- *B*(2)*E*(2) *V*(1)*Su*(0.5)*CC*(2)*F*(2)*Sa*(1) *BSo*(1) *BSu*(1)

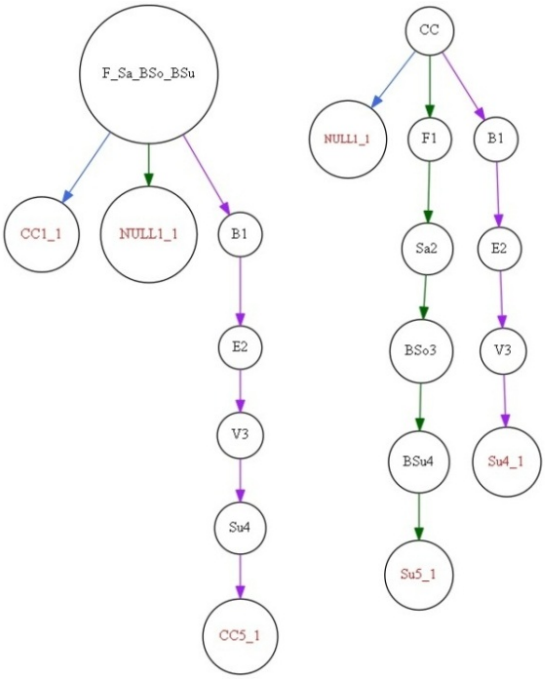


Fig. 5. Preconditions for Scenario 4 (behaviors *B*+*E*+*V*, *F*+*Sa*+*BSo*+*BSu*, and *CC*)

Scenario 5. This scenario aims to demonstrate that the system can successfully generalize problems of category 2:

Example 1. In a first setup, the learner is provided with the following demonstrations:

- *E*(2) *B*(2) *Su*(0.5) *BSu*(1) *F*(2)*BSo*(1) *N*(2) *Sa*(1) *V*(1) *CC*(2)
- *Sa*(1) *Su*(0.5) *E*(2) *CC*(2) *BSo*(1) *F*(2) *V*(1) *B*(2)*BSu*(1)
- *BSo*(1) *BSu*(1) *B*(2) *CC*(2) *E*(2)*F*(2)*Sa*(1) *Su*(0.5) *V*(1)

In this example, behavior *N* occurs in only one of the three sequences;this is lower than our threshold. Therefore *N*'s graph is removed from the set of possible behaviors and is not present in the execution sequence produced by the system:

- *E*(2) *B*(2) *Su*(0.5) *BSu*(1) *F*(2) *BSo*(1) *Sa*(1) *V*(1) *CC*(2)

Example 2. In a second setup, the learner is provided with these demonstrations:

- *E*(2)*B*(2) *Su*(0.5) *BSu*(1)*F*(2)*BSo*(1) *N*(2) *Sa*(1)*V*(1) *CC*(2)
- *Sa*(1)*Su*(0.5) *E*(2)*CC*(2) *BSo*(1)*F*(2)*V*(1) *B*(2) *BSu*(1)
- *BSo*(1) *N*(2) *BSu*(1) *B*(2)*CC*(2) *E*(2) *F*(2) *Sa*(1)*Su*(0.5) *V*(1)

In this example, behavior *N* appeared in two of the three training examples, and thus it is included as shown by the execution sequence being generated:

- *BSo*(1) *N*(2) *BSu*(1) *B*(2)*CC*(2)*E*(2)*F*(2)*Sa*(1) *Su*(0.5) *V*(1)

Scenario 6. This scenario aims to show that the system can identify and encode the types of situations illustrated by generalization category 1.c.iii:

- *CC*(2)*BSu*(1)*BSo*(1) *F*(2) *B*(2) *Su*(0.5) *Sa*(1) *V*(1) *E*(2)
- *V*(1)*CC*(2) *F*(2)*Sa*(1)*B*(2) *BSu*(1)*E*(2)*Su*(0.5)*BSo*(1)
- *Sa*(1) *Su*(0.5)*E*(2) *V*(1) *B*(2)*BSo*(1) *F*(2)*BSu*(1)*CC*(2)

In this case, the behavior add butter (*B*) was executed the fourth in each sequence in the example producing an execution sequence consistent with theexamples:

- *Sa*(1)*Su*(0.5) *E*(2)*V*(1)*B*(2) *BSo*(1) *F*(2) *BSu*(1) *CC*(2)

Scenario 7. This scenario aims to demonstrate that the system can successfully generalize problems of category 3:

Example 1. In the first setup, the learner is provided with these demonstrations:

- *Su*(0.5)*Sa*(1)*BSo*(1) *F*(2)*CC*(2)*St*(1) *St*(2)*St*(1) *St*(1)*BSu*(1) *V*(1)*B*(2)*E*(2) *St*(3)*St*(2)
- *BSu*(1)*BSo*(1) *E*(2) *Sa*(1) *CC*(2) *St*(1) *St*(1) *St*(1) *St*(1) *St*(1) *V*(1) *F*(2) *B*(2)*Su*(0.5) *St*(4) *St*(1)
- *B*(2)*V*(1)*E*(2)*BSu*(1) *Su*(0.5)*St*(2) *St*(2) *St*(1) *F*(2) *BSo*(1) *CC*(2) *Sa*(1) *St*(2) *St*(2) *St*(1)

Behavior *St* was repeated five times in two different places. Fig. 6 shows the precondition graph for *St*. The system detected the repeated behavior and at execution time it produced a sequence of *St* that used the most efficient way to stir 5 times:

- *BSu*(1)*BSo*(1) *E*(2) *Sa*(1) *CC*(2)*St*(4) *St*(1) *V*(1) *F*(2) *B*(2) *Su*(0.5) *St*(4) *St*(1)

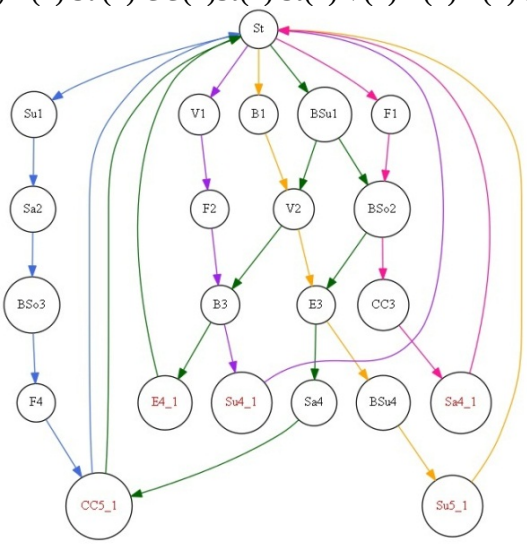


Fig. 6. Precondition Graph for Scenario 7 (Example 1: Behavior St)

Example 2. In the second setup, the learner is provided with these demonstrations:

- *Su*(0.5) *F*(1) *F*(1)*F*(0.5) *BSu*(1)*Sa*(1) *B*(2) *V*(1) *E*(2) *BSo*(1) *CC*(2)
- *E*(2) *BSo*(1)*Sa*(1)*B*(2)*F*(1)*F*(0.5)*F*(0.5)*F*(0.5)*BSu*(1)*V*(1) CC(2) Su(0.5)
- *CC*(2)*V*(1) *BSu*(1) *BSo*(1) *Su*(0.5) *E*(2) *F*(1)*F*(1)*F*(0.5) *Sa*(1)*B*(2)

In the second example the behavior *F* was repeated until there was 2.5 cups added in each sequence. The system detected the repeated behaviors and at execution time, produced a sequence whose post conditions met the training examples.

- *E*(2) *BSo*(1) *Sa*(1) *B*(2) *F*(2) *F*(0.5) *BSu*(1) *V*(1) *CC*(2) *Su*(0.5)

5. Discussion and future work

The experimental results presented above demonstrate that our system can successfully generalize the problems presented in Section 3. Furthermore the experiments establish that our system can successfully generate an execution sequence that is always consistent with the training samples.

However, our system could create a better execution sequence for the current environment if the state of the environment was recorded after a behavior was executed. This information would be used to figure out how many times a repeated behavior should occur.Along with this, future work also includes testing the proposed solution on a simulated and physical platform.

Acknowledgements. The work was supported by the National Science Foundation under award IIS-0546876.

References

1. Schaal, S. Is Imitation Learning the Route to Humanoid Robots? – Trends in Cognitive Sciences Vol. **6**, 1999, No 3, 242-323.
2. Voyles, R. M., J. D. Morrow, P. K. Khosla. Towards Gesture-Based Programming. – Robotics and Autonomous Systems, Vol. **22**, 1997, 361-375.
3. Voyles, R., P. Khosla. A Multi-Agent System for Programming Robots by Human Demonstration. – Integrated Computer-Aided Engineering, Vol. **8**, 2001, No 1, 59-67.
4. Pook, P. K., D. H. Ballard. Recognizing Teleoperated Manipulations. – In: Proc. Intl. Conf. on Robotics and Automation, 1993, 578-585.
5. Tominaga, H., J. Takamatsu, K. Ogawara, H, Kimura, K. Ikeuchi. Symbolic Representation of Trajectories for Skill Generation . – In: Proc., Intl. Conf. on Robotics and Automation, 2000, 4077-4082.
6. Ikeuchi, K., M. Kawade, T. Suehiro. Assembly Task Recognition with Planar, Curved and Mechanical Contacts. – In: Proc. IEEE Intl. Conf. on Robotics and Automation, 1993.
7. Kuniyoshi, Y., H. Inoue. Qualitative Recognition of Ongoing Human Action Sequences. – In: Proc., Intl. Joint Conf. on Artificial Intelligence, 1993, 1600-1609.
8. Avrahami-Zilberbrand, D., G. A. Kaminka. Fast and Complete Symbolic Plan Recognition. – In: Proc. Intl. Joint Conf. on Artificial Intelligence, 2005.
9. Abbas, T., B. A.MacDonald. Generalizing Topological Task Graphs From Multiple Symbolic Demonstrations in Programming by Demonstrations (PbD) Processes. – In: Proc. IEEE Intl. Conf. on Robotics and Automation, 2011, 3816-3821.
10. Konidaris, G., R. Grupen, A. Barto, S. Kuindersma. Robot Learning from Demonstration by Constructing Skill Trees. – The Intl. Journal of Robotics Research, Vol. **31**, 2012, 360-375.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0020

On Global Optimization of Walking Gaits for the Compliant Humanoid Robot, COMAN Using Reinforcement Learning

Houman Dallali, Petar Kormushev, Zhibin Li, Darwin Caldwell

Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova
Emails: houman.dallali@iit.it petar.kormushev@iit.it zhibin.li@iit.it Darwin.caldwell@iit.it

Abstract: *In ZMP trajectory generation using simple models, often a considerable amount of trials and errors are involved to obtain locally stable gaits by manually tuning the gait parameters. In this paper a 15 degrees of Freedom dynamic model of a compliant humanoid robot is used, combined with reinforcement learning to perform global search in the parameter space to produce stable gaits. It is shown that for a given speed, multiple sets of parameters, namely step sizes and lateral sways, are obtained by the learning algorithm which can lead to stable walking. The resulting set of gaits can be further studied in terms of parameter sensitivity and also to include additional optimization criteria to narrow down the chosen walking trajectories for the humanoid robot.*

Keywords: *Humanoid robot walking, compliance, reinforcement learning.*

1. Introduction

Walking trajectory generation for a humanoid robot is a challenging control problem. Humanoid robots have many Degrees of Freedom (DoF), with unstable, nonlinear and underactuated dynamics. Moreover, humanoid robots need to adapt their walking gait based on the environment, for instance to avoid an obstacle the robot needs to adapt its walking gait to take a different step size while keeping the same average speed.

Due to the complexity of the walking model several studies in the literature have approached the walking problem using machine learning techniques [1]. For example, in [2] a model-based reinforcement learning method was used for bipedal

walking on a planar 5 DoF robot fixed to a rotating boom. Learning was applied to learn the Poincare return map of the bipedal robot. The learning algorithm was used to minimize the torques while keeping a certain height to avoid falling. In [3] Central Pattern Generators (CPG) for QRIO humanoid were tuned using a policy gradient method and stable gaits were shown in simulation and in experiments. The robot's pelvis states were used to describe the motion of the robot. In [4] a stochastic policy gradient reinforcement learning was applied to a toy robot with a carefully designed passive mechanical dynamics and resulted in stable walking mainly due to the passive dynamics design. In [5] a map was constructed offline for a given trajectory to verify the feasibility of an step size. The set of all possible steps was considered to be a 6 dimensional space. The set was constrained to have a one to one correspondence between each trajectory and step size. Geometric and Zero Moment Point (ZMP) constraints where used to verify the feasibility of the desired walking step. In [6] reinforcement learning was used on a compliant bipedal legs to reduce the electrical energy consumption by changing the centre of mass of the robot during walking.

In terms of walking models, often simplified ones such as the inverted pendulum model [7-8] or the compass gait model [9] are used in trajectory generation and stability analysis while detailed and accurate models are used in simulation studies.

In addition to use of centre of mass, limit cycle criterion and capture point, one of the most commonly used stability criteria for humanoid walking is Zero Moment Point (ZMP) which is often formulated as a closed form solution of the linear inverted pendulum model of walking [10-11]. In a dynamically stable gait ZMP of a robot is the same as the centre of pressure.

Despite using simplified models for walking trajectory generation, a considerable effort in manual tuning is needed before the generated trajectory can be applied in practice on an actual humanoid robot. In order to address this problem, in this study an automatic way of tuning a walking gait using reinforcement learning methods is investigated. The goal of learning is to find dynamically stable gait parameters for a desired walking speed. In summary, reinforcement learning varies the walking gait parameters such as the step size and lateral sway amplitude to generate walking trajectories using the inverted pendulum model [8]. The produced gaits are verified in terms of stability walking speed using dynamic simulation of the compliant humanoid robot with 15 DoF. The simulation output, i.e. multibody stability and achieved walking speed are evaluated as a reward function for the learning algorithm to better tune the gait parameters. Applying this method on the robot is time consuming and can be risky, but in simulation we can obtain many stable solutions that can be further narrowed down using additional criteria such as energy efficiency. Hence, using the simulator the gaits are tested in advance, which creates a more reliable and feasible gait for the experiments.

In this paper, instead of using state-action based reinforcement learning which suffers from the curse of dimensionality, direct policy search reinforcement learning is used. This method works in a low dimensional policy space which bypasses the

dimensionality problem of the state-action space. The contributions of this study are twofold. Firstly, the method performs global search in the walking gait parameter space to yield multiple alternative solutions which can then be narrowed down using further optimization criteria such as energy. Secondly, it investigates the importance of different parameters on the walking gait.

This paper is organized as follows. In Section 2, the mechanical overview of COMAN humanoid robot and the dynamic walking simulator are described. In Section 3 the trajectory generation method is briefly described. The reinforcement learning algorithm is described in Section 4. The results of using this algorithm are presented in Section 5, and finally the conclusions and future work are briefly discussed.

2. Dynamic model of the humanoid robot COMAN

In this section, first, the mechanical description of the compliant humanoid robot COMAN is given. Next, an overview of the dynamic walking simulator and its features are discussed.

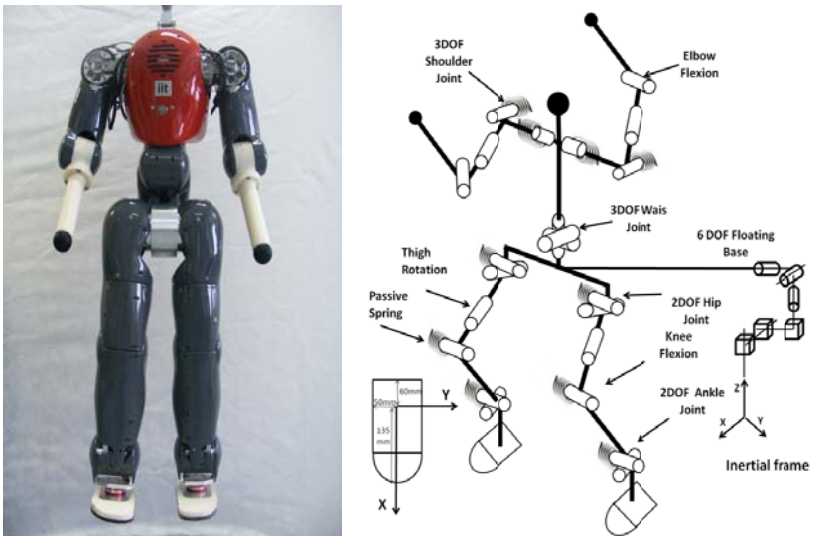
2.1. Overview of the mechanical model

COMAN (stands for COmpliant huMANoid) is powered by series elastic actuators and is being developed within the AMARSI European project [12] at the Italian Institute of Technology (IIT) as a derivative of the original iCub, and cCub [13] which added passive compliance in the major joints of the legs (see Fig. 1 (a)). The use of passive compliance provides shock protection, robust locomotion, safer interaction and potentially energy efficient locomotion. Currently, COMAN has 23 DoF, with passive compliance in the pitch joints in the legs, the waist, and the pitch and roll shoulder joints. In addition, COMAN uses brushless DC motors and harmonic drives, which are modeled in the dynamic simulator described in Subsection 2.2. In terms of control software architecture, currently COMAN uses a decentralized PID control architecture. Further details about the first prototype of COMAN, cCub are available in [13] with the major kinematic difference being the addition of passive compliance in the hip pitch and the orders of the ankle and the waist pitch and roll joints being swapped.

The coupled mechanical model of the robot is described in (1), where *M, C* and *G* are mass-inertia, Coriolis and gravity matrices. q, $\dot{q}$ and $\ddot{q}$ are positions, velocities and accelerations of all joints in vector form. Similarly, $q_m, \dot{q}_m$ and $\ddot{q}_m$ are the positions, velocities and accelerations of the motors. J and B_m are the motors' inertia and damping matrices. B_s and K_s are the passive compliance damping and stiffness matrices. τ_m is the motors' torque expressed in vector form,

$$\begin{cases} M\ddot{q} + C\dot{q} + Gq = B_s\left(\dot{q}_m - \dot{q}\right) + K_s\left(q_m - q\right) \\ J\ddot{q}_m + B_m\dot{q}_m + B_s\left(\dot{q}_m - \dot{q}\right) + K_s\left(q_m - q\right) = \tau_m \end{cases}. \quad (1)$$

In the next section, the dynamic walking simulator under Matlab is described.



(a) (b)

Fig. 1. COMAN humanoid robot (a) and the mechanical diagram with a floating base used to model the walking simulator (b)

2.2. Dynamic walking simulator

The dynamic walking simulator used in this paper considers 15 DoF of COMAN (legs and waist) and it is based on the floating base representation of legged robots as proposed in the literature [14-16]. A floating base is represented by additional six DoF attached between the humanoid robot and the world inertial frame to describe the free motion of the robot with respect to the inertial frame, as shown in Fig. 1 (b). The 6DoF consist of 3 translational joints and 3 rotational joints about the XYZ axis. This formalism unifies all the phases of legged locomotion, including a single support, double support and flight phase, as well as the fallen phase into one single model, and simplifies the simulation models by removing the switches between various phases of walking.

Since the floating base joints are not actuated, a realistic ground model must be introduced to simulate the robots balance and locomotion. The ground model is defined in the simulator using linear springs and dampers with limited friction to allow slipping on the floor. The impacts with the ground are modeled as compliant impacts as opposed to rigid body impacts. The tangential force of the ground is modeled in (2), where μ is the friction coefficient, and Δx is the deflection from the first point of contact. The normal force is given in (3), where F_z is the vertical ground reaction force, $\Delta z = z - z_0$ denotes the penetration in the ground which is always positive,

(2)
$$F_x = \begin{cases} -K\Delta x - D\Delta \dot{x}, & (-K\Delta x - D\Delta \dot{x}) \le \mu F_z \\ -\mathrm{sgn}(\Delta \dot{x})\mu F_z, & (-K\Delta x - D\Delta \dot{x}) > \mu F_z \end{cases},$$

(3) $$F_z = \begin{cases} 0 & z \geq z_0 \\ -K_G \Delta z - D_G \Delta \dot{z} & z < z_0 \end{cases}.$$

There are four points at each corner of the foot where the ground models are introduced as external forces as shown in Fig. 2. The parameters of the ground model are given in Table 1.

Table 1. Ground model coefficients

Symbol	Description	Value	Units
K_G	Vertical spring	150 000	N/m
D_G	Vertical damper	150	(N.s)/m
K_F	Friction stiffness	150 000	N/m
D_F	Friction damper	150	(N.s)/m
μ	Friction coefficient	0.9	⟨unitless⟩

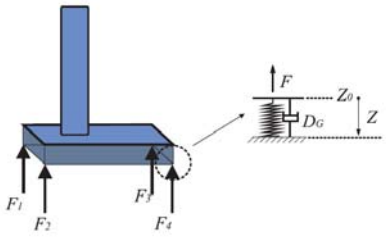


Fig. 2. Four ground contact point is introduced under each foot

Having developed a realistic simulation of walking for COMAN, the next section discusses the method used for walking trajectory generation.

3. Trajectory generation

A trajectory generator computes the reference trajectory for the robot's joints. This control system translates the desired walking parameters such as foot locations, step length, walking speed and walking direction into feasible and stable joints' trajectories. In this paper, the preview control method of ZMP based on the cart table model, proposed in [10], is used. The advantages of using this method are the ability of the robot to modify the reference trajectories according to the walking path and low computational cost which makes it suitable for online calculations. A brief description of this method is given below.

In this method, the cart-table linear model (Fig. 3) is used to formulate the relation between the centre of mass motion and the ZMP. This model has linear and decoupled dynamics in sagittal and lateral planes due to the constraint on the height of the centre of mass, which moves along a plane. This simplification in the nonlinear dynamics of an inverted pendulum results in derivation of the closed form equation between centre of mass and zero moment point $p_x = x - \frac{z_c}{g}\ddot{x}$, where x denotes the position of centre of mass and p_x is the position of the zero moment point in x direction. Since the dynamics are decoupled (constant height), the same equation holds for the y direction.

In order to generate the motion of the centre of the mass using (1) the initial position of the CoM and the ZMP should coincide. Based on the foot paths of the robot, a reference ZMP is derived as shown in Fig. 4 which shows the trajectory for three steps with 0.1 m step size. The continuous time based ZMP reference can be designed by using either linear or spline interpolation of a set of ZMP points with respect to time. The overall pattern generation scheme as well as the dynamic simulator and the learning algorithm are shown in Fig. 5. The objective locomotion parameters such as the walking speed and foothold planning are assumed to be given. Further details about the trajectory generator are provided in [10].

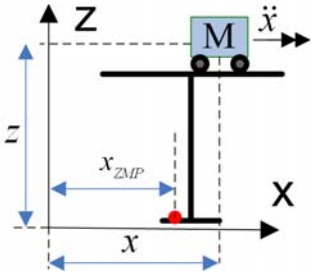


Fig. 3. Cart table model

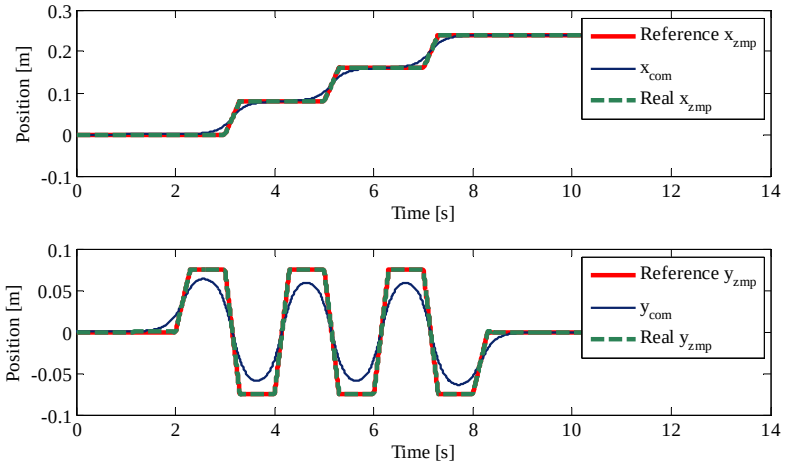


Fig. 4. The centre of mass and ZMP trajectories in sagittal and lateral planes

However, this method has a number of limitations. Firstly, the cart-table model only considers the overall CoM therefore the multi-body dynamics is not considered. Secondly, the control scheme assumes ideal position tracking and the dynamic effect of the springs in the compliant joints is not included. Therefore, the process of generating models using the simple cart table model and applying it to the full multibody system with compliance involves a considerable number of trial and errors. In other words, this method has only been applied before to find a locally stable gait, while for the first time in this paper, reinforcement learning for global search is proposed to explore the whole parameter space using the accurate dynamic walking simulation of COMAN. The result of the search yields multiple

stable solutions which can then be studied further for walking sensitivity analysis to gait parameter variations and also to include additional optimization criteria such as energy efficiency to choose among the dynamically feasible gaits. The learning algorithm is described in the next section.

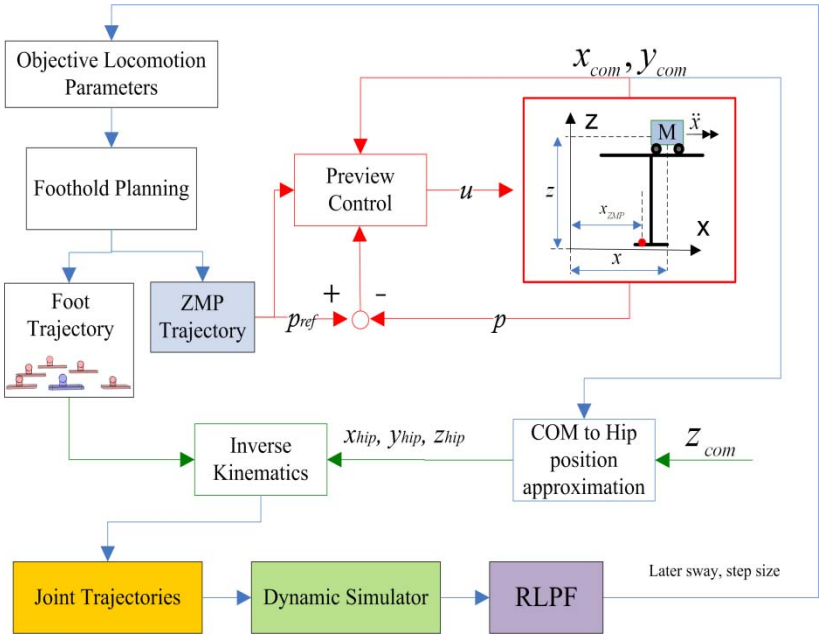


Fig. 5. Diagram of the overall learning based trajectory generation method is illustrated. The Cartesian position of the CoM of the robot is defined by x_{com}, y_{com} and z_{com}. The reference ZMP trajectory is p_{ref} and p is the ZMP feedback signal from the Cart table model which consists of x_{ZMP} and y_{ZMP}. The objective locomotion parameters namely, lateral sway and step size are provided by RLPF for a desired walking speed

4. Reinforcement learning

This section describes a recently proposed Reinforcement Learning algorithm based on Particle Filters (RLPF) for global search in policy space, which is capable of finding multiple alternative optimal policies [1]. The algorithm performs global search in the policy space, therefore eliminating the dependency on the policy initialization, and it has the ability to find the globally optimal policy.

Linking particle filters and RL is explained by the following observation. The landscape, defined by the reward function $R(\theta) \in \mathbb{R}$ over the whole continuous domain of the parameter space $\theta \in \Theta$, can be viewed as defining an Improper Probability Density Function (IPDF). An IPDF is similar to probability density function except that the integral of it does not have to be equal to one. This is possible even if the reward function $R(\theta)$ has negative values in its range, because a constant positive number can be added to the reward to obtain a non-negative reward function $R'(\theta)$ which has exactly the same set of optimizers $\theta^* \in \Theta$ as

$R(\theta)$. Hence, optimizing $R'(\theta)$ will also optimize $R(\theta)$. Based on this assumption that $R(\theta)$ is an IPDF, the RL problem can be reformulated as follows. Each trial $\tau(\pi(\theta))$ (τ denotes each trial and π is the policy for a given parameter vector θ) can be considered as an independent sample from the unknown IPDF. The RL algorithm chooses a finite number of sample points to find the values and modes of the unknown IPDF.

The main idea of RLPF is to use particle filtering as a method for choosing the sampling points, i.e. for calculating a parameter vector θ for each trial, which consist of the step size and lateral sway. A policy particle p_i is defined to be the tuple $p_i = \langle \theta_i, \quad \tau_i, \quad R_i \quad \omega_i \rangle$, where the particle p_i represents the outcome of a single trial τ_i performed by executing an RL policy $\pi(\theta_i)$, where θ_i is a vector of policy parameter values modulating the behavior of the RL policy π. The policy particle also stores the value of the reward function evaluated for this trial $R_i = R(\tau_i(\pi(\theta_i)))$. The variable τ_i contains task-specific information recorded during the trial depending on the nature of the task. The information in τ_i is used by the reward function to perform its evaluation. The variable ω_i is the importance weight of this policy particle, and the way of its calculation is explained as follows.

It is assumed that the set of particles $\{p_i\}$ is an approximate implicit representation of the underlying unknown IPDF defined by $R(\theta)$. Therefore, in order to select a new particle consistent with the real IPDF distribution, the samples are taken from the approximate distribution while correcting for the discrepancy. The mechanism for this correction is provided by the importance weights $\{\omega_i\}$.

Firstly, each policy particle p_i is assigned a scalar importance weight ω_i derived from its corresponding reward R_i using a transformation function g, such that $\omega_i \propto g(R_i)$. In the simplest case, $g(\cdot)$ could be the identity, but in the general case, it could be an arbitrary non-negative function. The function g is applied in such a way, that the importance weights are normalized, in the sense that $\forall \omega_i \; 0 < \omega_i < 1$, and also $\sum \omega_i = 1$. Secondly, an auxiliary function $h(u) = \int_{-\infty}^{u} \omega_u du$ is constructed, which takes the form $h(k) = \sum_{j=1}^{k} \omega_j$ in the discrete case. This function can be thought of as the (approximate) Cumulative Density Function (CDF) of the unknown PDF. Indeed, due to the way we create the importance weights, it follows directly that $\int_{-\infty}^{+\infty} \omega_u du = 1$, and thus $h(u)$ is a proper CDF. This is important because,

given that $\omega_i > 0$, it guarantees that $h(u)$ is strictly monotonically increasing and therefore the inverse function h^{-1} exists.

Algorithm 1 Reinforcement Learning based on Particle Filters (RLPF)

Input: parameterized policy π, policy parameter space Θ, reward function $R(\theta)$ where $\theta \in \Theta$, reward transformation function g, total number of trials N, initialization number of trials $L < N$, initial noise ε_0, noise decay factor λ, maximum number of particles σ.

Let $S = \{\}$ {A set of policy particles}
for $l = 1$ **to** L **do**
  Draw $\theta_l \sim U(\Theta)$ {Sample L initial particles}
  Perform trial $\tau_l(\pi(\theta_l))$
  Create new policy particle $p_l = \langle \theta_l, \tau_l, R_l, \omega_l \rangle$
  $S = S \cup \{p_l\}$
end for
for $n = L + 1$ **to** N **do**
  Let $h(0) = 0$
  for $i = 1$ **to** $|S|$ **do**
    $\omega_i = \frac{g(R_i)}{\sum_{j=1}^{|S|} g(R_j)}$ {Calc. importance weights}
    $h(i) = h(i-1) + \omega_i$ {Calc. aux. function}
  end for
  Draw $z \sim U(0,1)$
  Let $y = h^{-1}(z)$
  Let $k = \lceil y \rceil$ {$\lceil . \rceil$ is the ceiling function}
  Select policy particle $p_k = \langle \theta_k, \tau_k, R_k, \omega_k \rangle$
  Let $\varepsilon_n = \varepsilon_0 \lambda^{(n-L-1)}$ {noise with exp. decay λ}
  Let $\theta_n = \theta_k + \varepsilon_n$
  Perform trial $\tau_n(\pi(\theta_n))$
  Create new policy particle $p_n = \langle \theta_n, \tau_n, R_n, \omega_n \rangle$
  $S = S \cup \{p_n\}$
  if $|S| > \sigma$ **then**
    {Remove policy particle with smallest reward}
    $j = \arg\min_{p_j \in S} g(R_j)$
    $S = S \setminus \{p_j\}$
  end if
end for

Thirdly, a random variable z is introduced which is uniformly distributed in the interval (0, 1). Now, it can be shown that the random variable y defined as $y = h^{-1}(z)$ is distributed (approximately) according to the desired unknown PDF, see e.g. [17].

The goal of RLPF is not to approximate the expectation of a function, but rather, to find the mode (or modes) of the unknown function $R(\theta)$. The pseudo-code for RLPF is given in Algorithm 1. In the results section the reward function, number of trials, and gait parameter space are provided.

5. Simulation results

In this section, the results of applying RLPF algorithm to find optimal gait parameters for two desired walking speeds are presented. The reward function is defined in (4) where θ denotes the gait parameters, namely the step size and the lateral sway, B is a Boolean variable which is zero when the robot falls in the last trial and one otherwise, V_d is the desired average walking speed and V is the achieved walking speed in the dynamic simulator. The coefficients c_1 and c_2 are chosen to be 1000, and c_3 is 100. The goal of the reward function in (4) is to distinguish between the dynamically stable and unstable gaits and to feed back the achieved walking speed to the learning algorithm.

$$R(\theta) = c_1(1 - B) + c_2 e^{c_3 (V_d - V)^2}. \tag{4}$$

Initially the desired speed was set to 0.05 m/s and learning was used in 120 trials, where both a stable gait and the desired walking speed were achieved. The reward function is shown in Fig. 7. It can be seen that after the 30th trial the robot has not fallen since the reward is above 1000 points and the algorithm is only adjusting the walking speed. Moreover, the reward of each trial is color coded and plotted in Fig. 8 which shows that two clusters of parameters are found which gives the highest rewards (i.e., gaits which are stable and close to the desired walking speed). These two clusters of stable gaits have a vertical spread which suggests the lateral sway parameter has less sensitivity on stability compared to the step length. The lateral sway parameter is related to the dynamics of the robot in the lateral plane which has stiff joints (no passive compliance) and the step length parameter is directly related to the sagittal dynamics of the robot with passive compliance in the ankles, knees and the hips. Therefore, for a fixed passive compliance and using 120 trials a set of step sizes are derived which are between 1-3 cm. increasing the number of trials can further explore the parameter space and provide walking gaits with larger step sizes as shown in second experiment of this section. Also, this effect is due to the joint servo designs which are controlling the motors' positions to track the walking trajectories. Designing the servo controllers to control the link positions can improve the range of step sizes, while the achieved walking speed will depend on the bandwidth of the servo controllers. The robot will only be able to walk with trajectories speeds which are within its tracking bandwidth. The snapshot

of the optimal walking gait with the highest reward is shown in Fig. 6, where the robot takes four steps with speed of 0.0499 m/s.

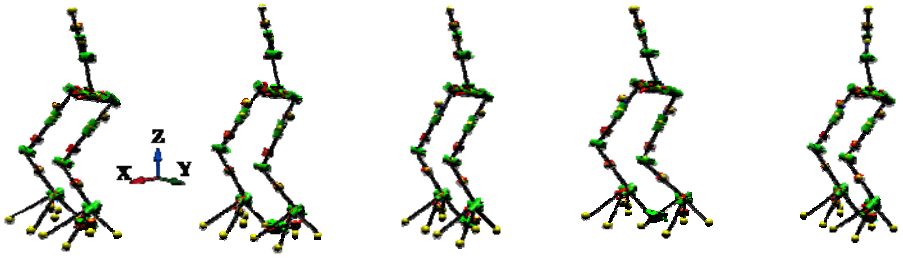


Fig. 6. Snapshots from the simulation of the gait computed in trial 120

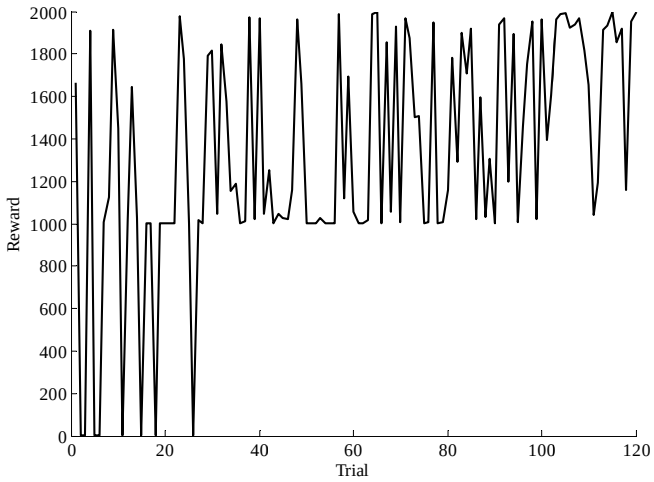


Fig. 7. Reward values during learning for desired speed of 0.05 m/s

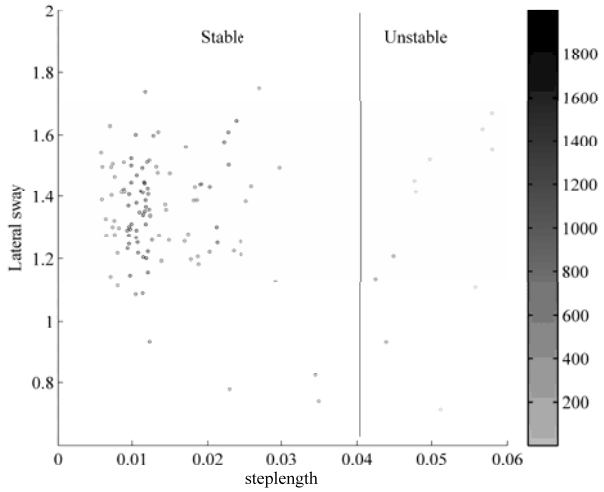


Fig. 8. Reward values in the walking gait parameter space with desired speed of 0.05 m/s

Moreover, the desired walking speed was set to 0.15 m/s and the RLPF algorithm was applied to 400 trials. This speed is above the tracking bandwidth of the current PID controllers set in the simulation and the robot achieved stable walking but the maximum achieved walking speed was lower (0.75 m/s).

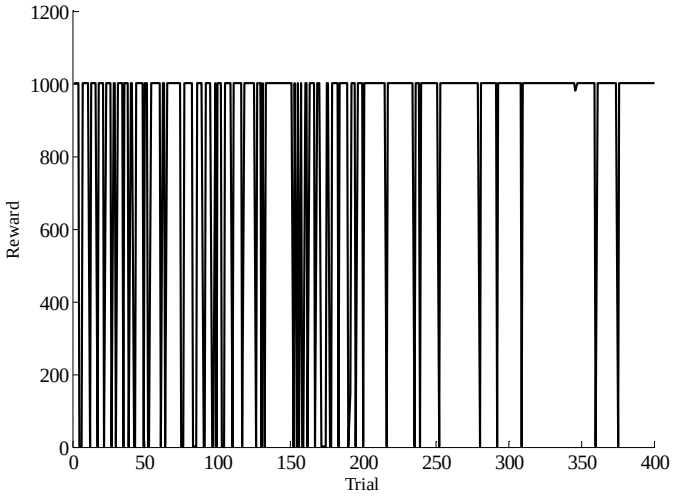


Fig. 9. Reward values during learning for desired speed of 0.15 m/s

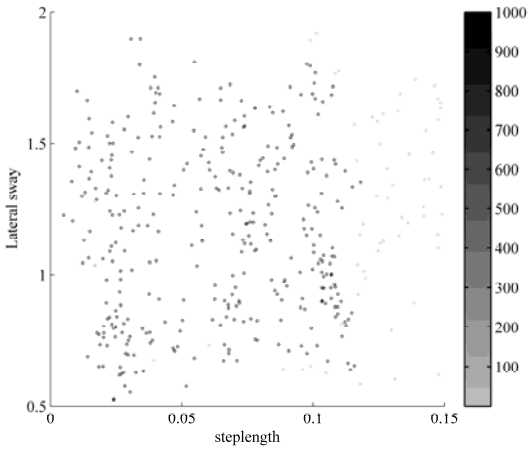


Fig. 10. Reward values in the walking gait parameter space with desired speed of 0.15 m/s

The reward values are shown in Fig. 9, which essentially distinguishes between the stable and unstable gaits, since the extra reward of walking close to the desired speed of 0.15 m/s is not obtained in the dynamic simulations. However, it can be seen that the falling frequency is decreasing with the number of trials and the learning is converging to higher rewards. The reward values in the parameter space are shown in Fig. 10, which shows the robot has taken steps sizes between 1-12 cm and has been stable.

6. Conclusion

In this paper the common problem of ZMP trajectory generation using simplified cart-table model which involves considerable amount of trials and errors to find locally stable gaits was considered. A reinforcement learning algorithm was combined with a dynamic walking simulator to perform automated global search for stable gaits in the walking parameter space. The algorithm was tested on two relatively low (0.05 m/s) and high (0.15 m/s) walking speeds and it found multiple sets of stable walking for a given speed with different step lengths and lateral sway. The result of this algorithm can be used to study the sensitivity of the gaits to parameter changes as well as including additional optimization criteria (such as energy efficiency) to narrow down the set of stable gaits.

In the future work, the designed walking gaits will be applied on the real robot, and in a possible future study to investigate the energy efficiency of a certain gait using simple models and learning methods to predict the optimal step length for a given walking speed. Also the simple reward function used in this study can be improved to better distinguish among different type of gaits.

Acknowledgements.: This work is supported by the FP7 European project AMARSi (ICT-248311).

References

1. Kormushev, P., D. G. Caldwell. Simultaneous Discovery of Multiple Alternative Optimal Policies by Reinforcement Learning. – In: IEEE International Conference on Intelligent Systems 2012, Sofia, Bulgaria (Accepted).
2. Morimoto, J., G. Cheng, C. G. Atkeson, G. Zeglin. A Simple Reinforcement Learning Algorithm For Biped Walking. – In: IEEE International Conference on Robotics and Automation'2004, New Orleans, LA, USA.
3. Endo, G., J. Morimoto, T. Matsubara, J. Nakanishi, G. Cheng. Learning CPG-Based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. – The International Journal of Robotics Research, Vol. **27**, 2008, No 2, 213-228.
4. Tedrake, R., T. W. Zhang, H. S. Seung. Stochastic Policy Gradient Reinforcement Learning on a Simple 3D Biped. – In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, Sendai, Japan. 2849-2854.
5. Wada, Y., K. Sumita. A Reinforcement Learning Scheme for Acquisition of Via-Point Representation of Human Motion. – In: IEEE International Joint Conference on Neural Networks'2004, Vols 1-4, 2004, 1109-1114.
6. Kormushev, P., B. Ugurlu, S. Calinon, N. G. Tsagarakis, D. G. Caldwell. Bipedal Walking Energy Minimization by Reinforcement Learning with Evolving Policy Parameterization. – In: IEEE/RSJ International Conference on Intelligent Robots and Systems'2011, San Francisco, CA, USA.
7. Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, H. Hirukawa. Biped Walking Pattern Generation by a Simple Three-Dimensional Inverted Pendulum Model. – Advanced Robotics, Vol. **17**, 2003, No 2, 131-147.
8. Kajita, S., F. Kanehiro, K. Kaneko, K. Yokoi, H. Hirukawa. The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation. – In: IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), 2001, 239-246.
9. Goswami, A., B. Thuilot, B. Espiau. A Study of the Passive Gait of a Compass-Like Biped Robot: Symmetry and Chaos. – International Journal of Robotics Research, Vol. **17**, 1998, No 12, 1282-1301.

10. Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa. Biped Walking Pattern Generation by Using Preview Control of Zero-Moment Point. – In: IEEE International Conference on Robotics and Automation (ICRA), 2003. 1620-1626.
11. Kajita, S., M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, K. Yokoi. Biped Walking Stabilization Based on Linear Inverted Pendulum Tracking. – In: IEEE/Rsj International Conference on Intelligent Robots and Systems (IROS), 2010.
12. AMARSI. Adaptive Modular Architectures for Rich Motor Skills. EU Supported FP7 Project. **http://www.amarsi-project.eu/**
13. Tsagarakis, G., N., Z. Li, J. Saglia, D. G. Caldwell. The Design of the Lower Body of the Compliant Humanoid Robot "cCub". – In: IEEE International Conference on Robotics and Automation (ICRA), 2011, Shanghai, China.
14. Buchli, J., M. Kalakrishnan, M. Mistry, P. Pastor, S. Schaal. Compliant Quadruped Locomotion Over Rough Terrain. – In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009, St. Louis, USA.
15. Featherstone, R. Rigid Body Dynamics Algorithms. 1st Ed. New York, Springer 2008, Science+Business Media, LLC. 280.
16. Mistry, M., J. Nakanishi, G. Cheng, S. Schaal. Inverse Kinematics with Floating Base and Constraints for Full Body Humanoid Robot Control. – In: IEEE-RAS International Conference on Humanoid Robots, 2008, Daejeon, Korea.
17. Bishop, C. M. Pattern Recognition and Machine Learning. New York, Springer, 2006.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0021

Combining Local and Global Direct Derivative-Free Optimization for Reinforcement Learning

Matteo Leonetti[1], *Petar Kormushev*[1], *Simone Sagratella*[2]

[1]*Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova*
[2]*Department of Computer, Control, and Management Engineering, Sapienza University of Rome, via Ariosto 25, 00185, Roma*
Emails: matteo.leonetti@iit.it petar.kormushev@iit.it sagratella@dis.uniroma1.it

Abstract: *We consider the problem of optimization in policy space for reinforcement learning. While a plethora of methods have been applied to this problem, only a narrow category of them proved feasible in robotics. We consider the peculiar characteristics of reinforcement learning in robotics, and devise a combination of two algorithms from the literature of derivative-free optimization. The proposed combination is well suited for robotics, as it involves both off-line learning in simulation and on-line learning in the real environment. We demonstrate our approach on a real-world task, where an Autonomous Underwater Vehicle has to survey a target area under potentially unknown environment conditions. We start from a given controller, which can perform the task under foreseeable conditions, and make it adaptive to the actual environment.*

Keywords: *Reinforcement learning, policy search, derivative-free optimization, robotics, autonomous underwater vehicles*

1. Introduction

Reinforcement Learning (RL) is the learning paradigm in which an agent improves its behavior on a given task by exploring the environment through trial and error. Its mathematical formulation consists in the maximization of a reward function, which measures the agent's performance. As an optimization problem it is a peculiar one, because many aspects of the task on which the performance must be optimized are

unknown to the agent. In particular, the function to be maximized has often no analytical expression, and must be sampled by acting in the environment. Each function evaluation costs the agent resources, which is especially critical in robotics. Robotic agents act in real-world environments, consume power, are subject to wearing and tearing, and work in real time. This imposes a careful use of trials, favoring on-line learning to off-line, batch, learning.

Policy gradient methods have largely been used in robotics [17] owing to characteristics that make them particularly suitable. In high-dimensional domains, finding good value function representations is difficult, while structuring the policy may come naturally from the task. Encoding the policy appropriately, policy search methods can benefit from previous knowledge. Usually fewer parameters are needed than with value function approximators, and the methods have strong theoretical foundations. Moreover, policies can be changed gradually, so that the behavior of the robot can be ensured withing the operation limits. Nonetheless, estimating the gradient is still a costly procedure, that requires many trials around the current policy. Noisy environments can make gradient estimation extremely difficult, either affecting the estimate greatly, or requiring a large number of trials.

While the local aspect of policy-gradient methods is favorable to robots, it is also a double-edged sword. On the one hand, policy-gradient approaches a local optimum slowly and smoothly, producing behaviors that don't deviate sharply from one another. On the other hand, there's no guarantee about the presence of global optima elsewhere. A broader exploration can be performed in simulation, but the so called *reality gap* [4, 11], the inevitable difference between the real and simulated domains, makes learning in simulation more brittle than in the actual environment.

In this paper, we explore direct derivative-free optimization algorithms for policy search in episodic reinforcement learning. Reinforcement learning imposes particular constraints on optimization algorithms. For instance, industrial applications are usually parallelized, while this is not possible for on-line RL. We combine the benefits of local methods for on-line learning, with a global search in simulation to obtain good starting points. While being able to benefit from all the advantages mentioned for policy-gradient algorithms, local derivative-free ones do not have the burden to estimate the gradient, which is particularly relevant with noisy reward functions. Derivative-free algorithms used in RL so far are pure global optimization algorithms, for the largest part evolutionary [1, 9]. We chose to combine two different derivative-free algorithms: a global stochastic search [3], and a line search [16]. The former allows to identify a policy in whose neighborhood the global optimum is most likely to be. The latter then refines this policy, going through its neighborhood without attempting to estimate the gradient. Learning in simulation is therefore coarse-grained, for the environment is only an approximation of the one the agent will really face. A more refined learning is performed on-line on the actual environment, in a local and smooth fashion. Different combinations of global and local methods are possible, and more sophisticated methods can be employed.

We implemented our approach on a real-world problem, where an Autonomous Underwater Vehicle (AUV) has to survey an area under unpredictable disturbances.

The AUV is equipped with a controller able to perform the task under normal operation conditions. When unexpected environment disturbances make it fail, the agent learns a policy that corrects the given controller. The experiments have been performed on a realistic simulator, taking into account real battery life. The method introduced in this paper proves to be able to learn a policy to perform the task in a few tens of trials, taking a time largely within the robot mission duration.

2. Background and notation

In this section we provide the background behind policy search in RL.

A Markov Decision Process is a tuple $MDP = \langle S, A, T, \rho \rangle$ where: S is a set of *states*, A is a set of *actions*, $T : S \times A \times S \to [0,1]$ is the transition function. $T(s,a,s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that the current state changes from s to s' by executing action a. $\rho : S \times A \times \mathbb{R} \to [0,1]$ is the reward function. $\rho(s,a,r) = Pr(r_{t+1} = r | s_t = s, a_t = a)$ is the probability to get a reward r from being in state s and executing action a. In our setting both states and actions are continuous, while time is discrete.

The behavior of the agent is represented as a function $\pi : S \times A \to [0,1]$ called a (stationary) *policy*, where $\pi(s,a)$ is the probability of selecting action a in state s. A policy π and an initial state s_0 determine a probability distribution d^π over the possible sequences $\omega = (\langle s_t, a_t, r_{t+1} \rangle, t \geq 0)$. Given such a sequence, we define the *cumulative discounted reward* as

$$R = \sum_{t \geq 0} \gamma^t r_{t+1}$$

where $0 < \gamma \leq 1$ is the *discount factor*. The reward is accumulated by executing a in s and following π thereafter.

A state is said to be *absorbing* if once entered cannot be left. An MDP with absorbing states is said to be *episodic*. In the rest of this paper we focus on episodic MDPs, where the duration of the episode is given by the first time step in which an absorbing state is entered. Therefore, from now on we set $\gamma = 1$ and sum over an unknown finite duration T.

Policies are parametrized through a vector θ. This allows to add structure to the policy, which is then changed by modifying the parameters. The value of a policy π is defined, through its parameters, as:

$$J(\theta) = \int_S d^\pi(s) \int_A \pi(s,a) r(s,a) ds da \tag{2}$$

where $r(s,a)$ is extracted from $\rho(s,a,\cdot)$.

Policy gradient methods try to estimate the gradient of the function in Equation 2, in order to make an update to the parameters along it:

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta_i) \tag{3}$$

Direct derivative-free algorithms, on the other hand, perform hill-climbing updates without estimating the gradient. The function our method will maximize, over a finite horizon and from a single initial state, is:

$$J(\theta) = \sum_{t=0}^{T} \int_S d_t^{\pi}(s) \int_A \pi(s,a) r(s,a) ds da \tag{4}$$

3. Related work

A large number of policy search methods have been developed in the field, although only few of them have found successful application in robotics [14]. Most of those are local algorithms, the vast majority of which is gradient-based. Episodic REINFORCE [26] is one of the earliest gradient-based learning algorithm for episodic RL. As previously mentioned, gradient-based methods estimate the gradient at a given policy, and perform an update along its direction according to Equation 3. More recent policy gradient [22] RL algorithms are G(PO)MDP [2], natural policy gradient [12], and natural actor-critic [19] methods. All of the mentioned policy gradient methods differ in how the gradient is estimated. A different approach, based on Expectation Maximization (EM), has been proposed and applied to several algorithms, the most recent of which is PoWER [18, 14]. In EM-based methods, the lower bound on the expected cumulative return is maximized. Another notable local algorithm is Policy Improvements with Path Integrals (PI^2) [23], which belongs to the category of path integrals method applied to stochastic optimal control.

The algorithm we are going to apply for local search is a line search, as in gradient-based methods, but the direction used is not the gradient. Therefore, it can avoid the extremely costly operation of estimating it. Nonetheless, the algorithm is guaranteed, under appropriate conditions, to converge to the optimal solution.

On the side of global algorithms, a huge range of methods has been employed. Simulated Annealing [13] and Ant Colony Optimization [6] are well-known global methods, although the most studied one is the set of evolutionary approaches, in particular genetic algorithms [8]. Tabu search [7] attempts to escape local minima by performing temporary worsening moves, and using history to prevent steps that would lead back to the minimum just left. Last, we mention RLPF [15] and pattern search [24] belonging to the class of global random optimization algorithms used in RL. The global algorithm we are going to use is a clustering algorithm similar to pattern methods, where the search is random and controlled. In our random search, an initially random point cloud tends to gather around the global optimum, as will be described in Section 4.1.

4. Derivative-free algorithms

It is well known that extensive useful information is contained in the derivatives of any function one wishes to optimize. For a variety of reasons, however, there have

always been many instances where (at least some) derivatives are unavailable or unreliable. Nevertheless, under such circumstances it may still be desirable to carry out optimization. Consequently, a class of nonlinear optimization techniques called derivative-free methods has been developed in the literature. In the following, we describe the methods we are going to combine, highlighting the features that make them suitable for the different learning phases. The first method, a random search, was developed in the original paper with both a global and a local search. We removed the local part, and substituted it with the line search described in Section 4.2. This line search fits better with the RL setting, according to the arguments on locality introduced in Section 1. In the following, the algorithms will be described for minimization problems, in order to comply with the literature on optimization. Since in our RL implementation we maximize the reward (instead of minimizing costs) the sign of the reward function will simply be inverted.

4.1. Random global serach

Many algorithms have been proposed in the literature to solve unconstrained global optimization problems [10, 25]. In this paper, however, we are interested to tackle the particular difficult case of a problem in which:

- the evaluation of the objective function is very expensive;
- the values of the objective function can be affected by the presence of noise;
- the derivatives of the objective function are not available.

The method we use is a version of the Controlled Random Search Algorithm [20] in which the efficiency is improved by using a weighted centroid and a weighted reflection [3].

4.1.1. Controlled Random Search Algorithm

Data: a positive integer $m \geq n+1$, $\varepsilon > 0$

Step 0. Set $k = 0$ and compute the initial set:

$$S^k = \{\theta_1^k, \ldots, \theta_m^k\}$$

where the points θ_i^k, $i = 1, \ldots, m$ are chosen at random over a box D; evaluate J at each point θ_i^k, $i = 1, \ldots, m$.

Step 1. Determine the points θ_{max}^k, θ_{min}^k and the values J_{max}^k, J_{max}^k such that:

$$J_{max}^k = J(\theta_{max}^k) = \max_{\theta \in S^k} f(\theta)$$
$$J_{min}^k = J(\theta_{min}^k) = \min_{\theta \in S^k} f(\theta);$$

if $J_{max}^k - J_{min}^k \leq \varepsilon$ then STOP

Step 2. Choose at random $n+1$ points $\theta_{i_0}^k, \theta_{i_1}^k, \ldots, \theta_{i_n}^k$ over S^k, where

$$J(\theta_{i_0}^k) \geq J(\theta_{i_j}^k), \quad j = 1, \ldots, n;$$

determine the centroid

$$c^k = \sum_{j=0}^{n} w_j^k \theta_{i_j}^k,$$

and determine the trial point $\bar{\theta}^k$ given by

$$\bar{\theta}^k = c^k - \alpha^k(\theta^k_{i_0} - c^k),$$

where

$$w^k_j = \frac{\eta^k_j}{\sum_{r=0}^n \eta^k_r}, \quad \eta^k_j = \frac{1}{J(\theta^k_{i_j}) - J^k_{min} + \phi^k}, \quad \alpha^k = 1 - \frac{J(\theta^k_{i_0}) - \sum_{j=0}^n w^k_j J(\theta^k_{i_j})}{J^k_{max} - J^k_{min} + \phi^k},$$

and

$$\phi^k = n\frac{(J^k_{max} - J^k_{min})^2}{J^0_{max} - J^0_{min}};$$

if $\bar{\theta}^k \notin D$ go to Step 2; otherwise compute $J(\bar{\theta}^k)$.

Step 3. If $J(\bar{\theta}^k) \geq J^k_{max}$ then take

$$S^{k+1} = S^k;$$

set $k = k+1$ and go to Step 2

Step 4. If $J(\bar{\theta}^k) < J^k_{max}$ then take

$$S^{k+1} = S^k \cup \{\bar{\theta}^k\} - \{\theta^k_{max}\};$$

set $k = k+1$ and go to Step 1

The initial population of points is expected to cluster in the region where the global optimum is most likely to be. In practice, the possibility of locating a global minimum point rests on the fact that the number of points randomly chosen at the initial step is not small, and that global minimum points do not have narrow region of attraction. From this point of view, it appears clearly that the described algorithm is a heuristic. In this respect, we refer to section 7.2 of Torn and Zilinskas' book [25], where it is reported a thorough discussion on why heuristics are necessary and on how many the heuristics elements in global optimization are.

4.2. Local refinement

Numerical experience seems to indicate that the algorithm described in Section 4.1 is efficient enough at the global search, while it is not able to perform a sufficiently fast local minimization when the algorithm has produced an estimate $\tilde{\theta}$ "good enough". A local refinement procedure seems to be a better way to compute the global minimizer when a point $\tilde{\theta}$ near the global minimum is provided by the global method. In fact derivative-free local methods have better convergence properties than global ones and the risk to compute a local minimizer that is not global is unlikely by starting the local method from $\tilde{\theta}$.

In this paper, for the local refinement, we use a direct-search method that is a line-search version of the coordinate-search algorithm. Direct-search methods are derivative-free methods that sample the objective function at a finite number of points. At each iteration, they decide which actions to take solely based on those function

values, and without any explicit or implicit derivative approximation or model building [5]. Two particular subclasses of globally convergent direct-search methods are pattern-search methods and line-search methods. Pattern-search methods present the distinguishing feature of evaluating the objective function on specified geometric patterns. Line-search methods, on the contrary, draw their inspiration from the gradient-based minimization methods and perform one dimensional minimization along suitable directions. These two classes of methods present different interesting features. Pattern search methods can accurately sample the objective function in a neighborhood of a point. Hence, they can identify a "good" direction, namely, a direction along which the objective function decreases significantly. Line search algorithms can perform large steps along the search directions and, therefore, can exploit to a large extent the possible goodness of the directions. The algorithm used here combines these approaches in order to determine "good" directions and to perform "significant" step lengths along such directions [16].

4.2.1. Coordinate-Search Algorithm With Line-Search Expansions

Data: $\theta^0 \in \mathbb{R}^n$, $\tilde{\alpha}_1^0, \ldots, \tilde{\alpha}_n^0 > 0$, $\sigma \in (0,1)$, $\gamma \in (0,1)$, $\delta > 1$, $\varepsilon > 0$

Step 0. Set $k = 0$

Step 1. If $\max_{i=1,\ldots,n} \{\tilde{\alpha}_i^k\} \leq \varepsilon$ then STOP; set $i = 1$, $y_1^k = \theta^k$, $x^k = \theta^k$

Step 2. If $J(y_i^k + \tilde{\alpha}_i^k e_i) \leq J(y_i^k) - \gamma(\tilde{\alpha}_i^k)^2$ and $J(y_i^k + \tilde{\alpha}_i^k e_i) < J(x^k)$ then
set $\alpha_i^k = \tilde{\alpha}_i^k$ and $x^k = y_i^k + \alpha_i^k e_i$
while $J(y_i^k + \delta\alpha_i^k e_i) \leq J(y_i^k) - \gamma(\delta\alpha_i^k)^2$ and $J(y_i^k + \delta\alpha_i^k e_i) < J(x^k)$
set $\alpha_i^k = \delta\alpha_i^k$ and $x^k = y_i^k + \alpha_i^k e_i$
end while
set $\tilde{\alpha}_i^{k+1} = \alpha_i^k$
else set $\alpha_i^k = 0$ and $\tilde{\alpha}_i^{k+1} = \sigma\tilde{\alpha}_i^k$

Step 3. Set $y_{i+1}^k = y_i^k + \alpha_i^k e_i$

Step 4. If $i < 2n$ then set $i = i + 1$ and go to Step 2

Step 5. Set $\theta^{k+1} = x^k$, $k = k + 1$ and go to Step 1

5. Experimental evaluation

We carried out our experiments on a simulator of the vehicle Girona500 [21]. Girona-500 is a reconfigurable autonomous underwater vehicle designed for a maximum operating depth of up to 500 m. The vehicle has passive stability in pitch and roll, making it suitable for imaging surveys. The most remarkable characteristic of Girona 500 is its capacity to reconfigure for different tasks. On its standard configuration, the vehicle is equipped with typical navigation sensors (DVL, AHRS, pressure gauge and USBL) and a basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor). In addition to these sensors, almost half the volume of the lower hull is reserved for mission-specific payload such as a stereo imaging system or an electric arm for manipulation tasks. The same philosophy has been applied to the propulsion system. The basic configuration has 4 thrusters, two vertical to actuate the heave and pitch and two horizontal for the yaw and surge. However, it is

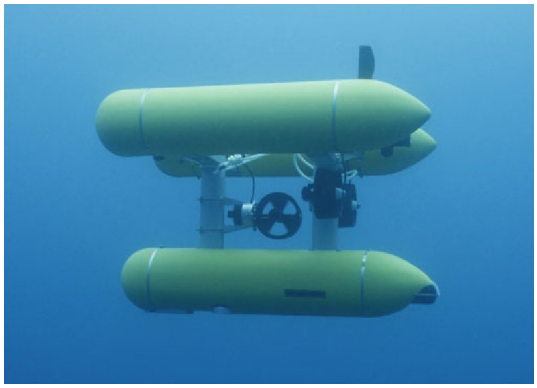


Fig. 1. The robot Girona500

possible to reconfigure the vehicle to operate with only 3 thrusters (one vertical and two horizontal) and with up to 8 thrusters to control all the degrees of freedom.

We performed our experiment using the Under Water Simulator (UWSim)[1] and the Robot Operating System (ROS)[2].

The robot's mission is a survey task, in which the agent has to reach a black box on the bottom of the seabed to take images and compose a mosaic. The target of the robot is to stay within the distance of 1m from the given point as long as possible, despite the disturbances in the environment. A current with velocity 0.6m/s has been simulated at depth higher than 2m, while the target point is 4.8m deep. Therefore, the agent cannot avoid the current and has to navigate trough it. The setting is represented in Figure 2. The current is too strong to be fully compensated

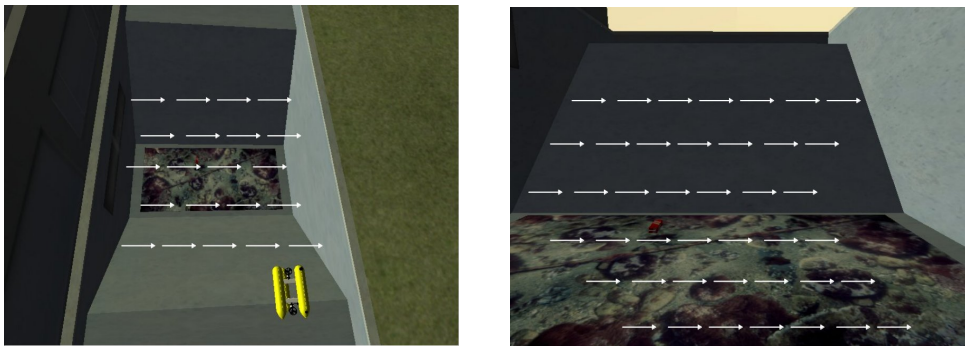


Fig. 2. The experimental setting. A strong current pushes the vehicle below 2m depth

by the thrusters, and once entered the robot slowly drifts away from the target point. We assume the agent is provided with an initial controller, able to perform the task

[1] http://www.irs.uji.es/uwsim/

[2] http://www.ros.org

under normal operation conditions. In our experiment this was a simple PD controller. The trajectory produced by the given controller is shown in Figure 3.

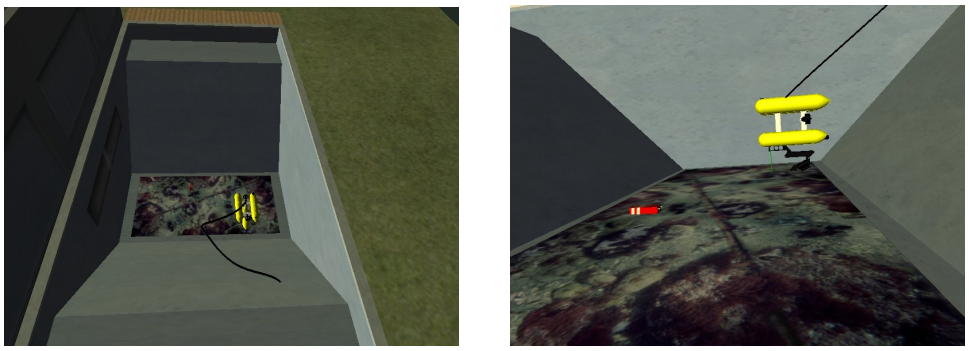


Fig. 3. The initial trajectory, produced by the PD controller alone, without learning.

Any robust controller can react to the environment but cannot counteract in advance unpredictable disturbances. Once the robot has entered the current there is little to do other then slow down the drifting. Therefore, the trajectory must be modified *before* entering the current, foreseeing its effect and counteracting them in advance. This is only possible by exploring the environment and learning about its characteristics. It requires to face the same task several times to make use of past experience to predict actions' effects. This is the focus of this paper: making a given controller adaptive and able to perform a task on which it would otherwise fail.

We assume an episodic scenario where the agent is able to return to the initial location (which is not inside the current) and start again. The battery power of Girona500 allows for 5 to 6 hours of operation, therefore we allocate for the on-line learning of behaviors no longer than 1 hour.

The reward function is given by:

$$J(\theta) = \begin{cases} -d & \text{if } d \geq D \\ t_D - D & \text{if } d < D \end{cases}$$

where D is a distance threshold, d is the minimum distance reached from the target, and t_D is the time spent at a distance from the target lower than D. In our experiments, $D = 1\text{m}$. Both d and t_D depend on θ through the policy, as expressed in Equation 4. Intuitively, the agent tries to maximize the time spent within 1m from the given point, using the distance as a heuristic when it is far from the target.

The policy computed and optimized is a correction to the given controller, whose output is velocity. Therefore, the actions are the continuous velocities summed to the controller's one. The policy has 18 parameters, and is a linear combination of position and velocity along the three axes for each component of the action.

5.1. Results

We first ran the global algorithm with an initial population of 100 samples. This phase is an off-line learning, and its aim is to provide a good starting point for the local search. Since, in this paper, we are going to run the second phase in simulation as well, we account for the discrepancy between the simulator and the actual environment by having a slightly different current in the two phases. The results are shown in Figure 4. After a few hundred trials, the algorithm finds a point above the threshold,

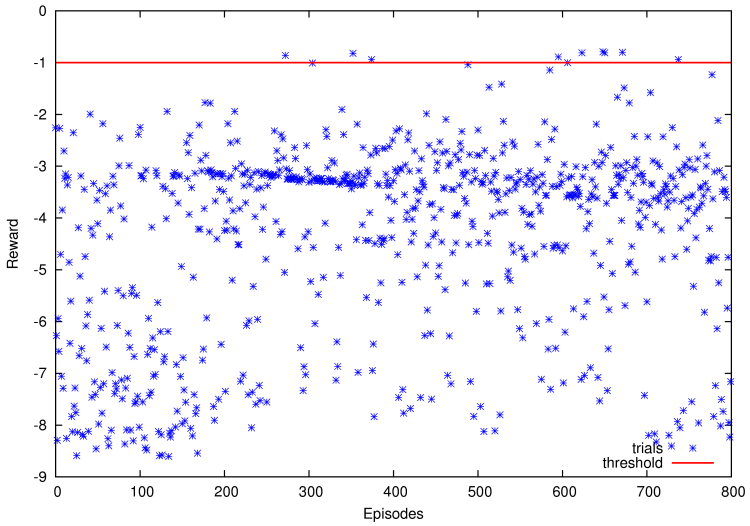


Fig. 4. The results of the trials during the initial random search

that is where the agent is able to approach the target point closer than 1m.

We took the best point found in this phase, and used it to initialize the policy for the local search. In this second phase, the current is the *real* one, which without learning produces the trajectory in Figure 3. We limited learning to 150 episodes, which take about one hour of real time. The policy learned during the first phase, when applied to the current of the second phase produces the trajectory shown in Figure 5. It manages to reach the point by staying higher than the current for as long as possible, and then entering it. Although learned under slightly different environment conditions, this behavior is already a good one. At this stage the agent is able to get within 1m from the target, therefore being able to perform the survey task. Local learning will attempt to maximize the time spent in this area, despite the real current.

The results of the local search are shown in Figure 6. The plot shows the learning curves from both the initial policy given from the first phase, and from the zero vector, which corresponds to no correction to the given controller. The local algorithm proved fast enough to reach the desired zone (-1 threshold in the reward) within the hour allocated for on-line learning. This shows that even with no prior information, this version of line search is well suited for Reinforcement Learning. Moreover, starting from the policy obtained through global learning, it performed better from

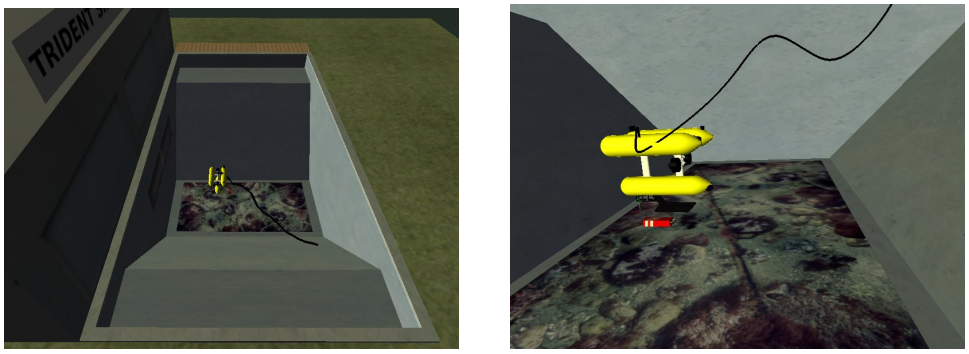


Fig. 5. The trajectory obtained by the first phase of learning, with the current of the real environment. The current in this picture is different from the one during learning.

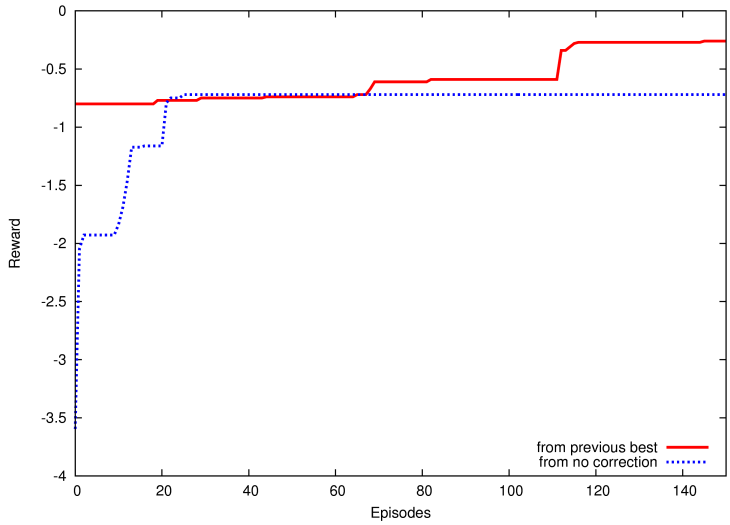


Fig. 6. Results of local learning from the initial point determined during the first phase and from the zero vector, which corresponds to no correction.

the first trials, reaching a even higher reward. The global random search, therefore, has been able to find a point with a better neighborhood than the natural initial point, that is when no correction is applied to the given controller.

6. Conclusion

The two algorithms we combined fit well with the paradigm of reinforcement learning in robotics. Simulators are fundamental tools, which are often used in place of the real robot for dangerous exploration. Off-line learning, however, cannot alone achieve adaptivity in unknown environments and go beyond what has been modeled

in the simulator. We showed how derivative-free line-search algorithms can be effective in reinforcement learning, and provide an alternative to gradient-based methods while retaining most of the features that made them so popular in robotics. With the two-phased learning strategy presented in this paper, we tailored an optimization algorithm for both off and on-line learning, obtaining the best from the global and the local method. Notably, this provides a very general methodology to wrap a learning framework around a given controller, increasing its robustness and ability to adapt the environments, especially under unmodeled circumstances.

7. Acknowledgments

This work was supported by the PANDORA European project under contract FP7-ICT-288273.

References

1. Barash, D .. A genetic search in policy space for solving Markov decision processes, – In: AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information, 1999.
2. Baxter, J., P. Bartlett. Direct gradient-based reinforcement learning, – In: Proceedings of IEEE International Symposium on Circuits and Systems, Vol. 3, IEEE, 2000, 271–274.
3. Brachetti, P., M. De Felice Ciccoli, G. Di Pillo, S. Lucidi. A new version of the Price's algorithm for global optimization, Journal of Global Optimization, Vol. 10, No 2, 1997, 165–184.
4. Carpin, S., M. Lewis, J. Wang, S. Balakirsky, C. Scrapper. Bridging the gap between simulation and reality in urban search and rescue, Robocup 2006: Robot Soccer World Cup X, 1–12.
5. Conn, A., K. Scheinberg, L. Vicente. Introduction to derivative-free optimization, Vol. 8, Society for Industrial Mathematics, 2009.
6. Dorigo, M., M. Birattari, T. Stutzle. Ant colony optimization, IEEE Computational Intelligence Magazine, Vol. 1, No 4, 2006, 28–39.
7. Glover, F., M. Laguna. Tabu search, Vol. 1, Springer, 1998.
8. Goldberg, D .. Genetic algorithms in search, optimization, and machine learning, Addison-wesley, 1989.
9. Gomez, F., J. Schmidhuber, R. Miikkulainen. Efficient Non-Linear Control through Neuroevolution, – In: Proceedings of the European Conference on Machine Learning, Springer, Berlin, 2006, 654–662.
10. Horst, R., P. Pardalos, N. Thoai. Introduction to global optimization, Springer, 2000.
11. Jakobi, N., P. Husbands, I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics, Advances in artificial life, 704–720.
12. Kakade, S .. A natural policy gradient, Advances in neural information processing systems, Vol. 14, 2001, 1531–1538.
13. Kirkpatrick, S., C. Gelatt Jr, M. Vecchi. Optimization by simulated annealing, Science, Vol. 220, No 4598, 1983, 671–680.
14. Kober, J., J. Peters. Policy search for motor primitives in robotics, Machine learning, Vol. 84, No 1, 2011, 171–203.
15. Kormushev, P., D. G. Caldwell. Simultaneous Discovery of Multiple Alternative Optimal Policies by Reinforcement Learning, – In: IEEE International Conference on Intel-

ligent Systems (IS 2012), 2012.
16. Lucidi, S., M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization, SIAM Journal of Optimization, Vol. 13, No 1, 2002, 97–116.
17. Peters, J., S. Schaal. Policy gradient methods for robotics, – In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, 2219–2225.
18. Peters, J., S. Schaal. Reinforcement learning by reward-weighted regression for operational space control, – In: Proceedings of the 24th international conference on Machine learning, ACM, 2007, 745–750.
19. Peters, J., S. Vijayakumar, S. Schaal. Natural Actor-Critic, – In: Proceedings of the 16th European Conference on Machine Learning (ECML), 2005, 280–291.
20. Price, W.. Global optimization by controlled random search, Journal of Optimization Theory and Applications, Vol. 40, No 3, 1983, 333–348.
21. Ribas, D., N. Palomeras, P. Ridao, M. Carreras, A. Mallios. Girona 500 AUV, from survey to intervention, IEEE/ASME Transactions on Mechatronics, Vol. 17, No 1, 2012, 46–53.
22. Sutton, R., D. McAllester, S. Singh, Y. Mansour. Policy gradient methods for reinforcement learning with function approximation, Advances in neural information processing systems, Vol. 12, No 22.
23. Theodorou, E., J. Buchli, S. Schaal. A generalized path integral control approach to reinforcement learning, The Journal of Machine Learning Research, Vol. 9999, 2010, 3137–3181.
24. Torczon, V., et al.. On the convergence of pattern search algorithms, SIAM Journal on optimization, Vol. 7, No 1, 1997, 1–25.
25. Torn, A., A. Zilinskas. Global Optimization, Springer, 1989.
26. Williams, R.. Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning, Vol. 8, No 3, 1992, 229–256.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0022

Learning Fast Quadruped Robot Gaits with the RL PoWER Spline Parameterization

Haocheng Shen, Jason Yosinski, Petar Kormushev, Darwin G. Caldwell and Hod Lipson

Cornell University, 239 Upson Hall, Ithaca, NY 14853, USA
Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova, Italy
Emails: hs454@cornell.edu yosinski@cs.cornell.edu Petar.Kormushev@iit.it

Abstract: *Legged robots are uniquely privileged over their wheeled counterparts in their potential to access rugged terrain. However, designing walking gaits by hand for legged robots is a difficult and time-consuming process, so we seek algorithms for learning such gaits to automatically using real world experimentation. Numerous previous studies have examined a variety of algorithms for learning gaits, using an assortment of different robots. It is often difficult to compare the algorithmic results from one study to the next, because the conditions and robots used vary. With this in mind, we have used an open-source, 3D printed quadruped robot called QuadraTot, so the results may be verified, and hopefully improved upon, by any group so desiring. Because many robots do not have accurate simulators, we test gait-learning algorithms entirely on the physical robot. Previous studies using the QuadraTot have compared parameterized splines, the HyperNEAT generative encoding and genetic algorithm. Among these, the research on the genetic algorithm was conducted by (G l e t t e et al., 2012) in a simulator and tested on a real robot. Here we compare these results to an algorithm called Policy learning by Weighting Exploration with the Returns, or RL PoWER. We report that this algorithm has learned the fastest gait through only physical experiments yet reported in the literature, 16.3% faster than reported for HyperNEAT. In addition, the learned gaits are less taxing on the robot and more repeatable than previous record-breaking gaits.*

Keywords: *Evolvable splines, parameterized gaits, HyperNEAT, machine learning, quadruped.*

1. Introduction

Various learning algorithms have been proved to be effective for legged robots. Algorithms, such as HyperNEAT (Y o s i n s k i et al. [12]), Genetic Algorithms (C h e r n o v a and V e l o s o [10]) and others (H o r n b y et al. [7]; Z y k o v et al. [13]; T e´ l l e z et al. [2]; V a l s a l a m and M i i k k u l a i n e n [11]) have been tested to be effective for automatic learning gaits for robots. Despite competitive performance, a major task is usually hidden in the published results: tuning the parameters for these evolutionary algorithms (K o r m u s h e v et al. [9]). Here we present results using a different way of learning gaits: a Reinforcement Learning algorithm called Policy learning by Weighting Exploration with the Returns, or RL PoWER, proposed by K o b e r and P e t e r s [8]. In our experiment, the main focus of the research is on the applicability of RL PoWER to quadruped robot gait learning. Another motivation is to compare the state-of-the-art neural network algorithm, HyperNEAT, with our proposed method in quadruped robot gait learning.

Fig. 1. The open source, 3D printed QuadraTot robot used in this research. The white printed booties are new additions to prevent sliding on surfaces and to minimize measurement error

2. Problem definition

As in Yo s i n s k i et al. [12], we define the gait learning problem to be the search for a gait that maximizes some specific metric. Mathematically, we define a gait as a function that specifies a vector of commanded motor positions for a robot over time. Gaits without a feedback – also called open-loop gaits, can be defined as

$$\boldsymbol{x} = g(t)$$

According to this definition, open-loop gaits are deterministic. One particular gait should behave exactly the same when it is run from a trial to a trial. However, the actual robot motion and fitness measured will vary due to the errors and uncertainty of the real world physics. In our trials, the gaits generated were sent to the robot and executed in an open loop manner. We will measure and analyze the performance between HyperNEAT and RL PoWER, the latter of which will be the focus of discussion in this paper. The metric used for the fitness in this paper will be described later.

3. Related work

Many attempts have been made on robot gaits learning using machine learning algorithms, producing competitive results (Chernova and Veloso [2]; Hornby et al. [7]; Zykov et al. [13]; Clune et al. [3, 4]; Tellez et al. [10]; Valsalam and Miikkulainen [11]). In fact, Jason et al. tested and compared six different algorithms for quadruped robot gait learning in their studies. HyperNEAT, a generative encoding neural network algorithm, achieved the best performance among all six methods. A following research done by Kyrre et al., using a tuned simulator, generated even more competitive results. In Kormushev et al. [9] bipedal robot energy reduction research, a reinforcement learning algorithm was used to optimize the performance. The results from Kormushev's research showed this algorithm's potential for walking problems. This reinforcement learning method still needs more tests for explorations.

4. Methods

4.1. Policy representation by splines

The simplest model with back-compatibility is geometric splines. For example, for a given model $f(x)$ with K knots, we can preserve the exact shape of the generated curve while adding extra knots to the original spline. If we put one additional knot between every two consecutive knots of the original spline, we end up with a $2K - 1$ knots and a spline that has the same shape as the original one. In order to do this, we need to define an algorithm for evolving the parameterization from K to L knots ($L > K$), which is formulated as in 1. Without loss of generality, the policy parameters are normalized into [0, 1], and appropriately scaled and shifted as necessary upon use.

Algorithm 1. EvolvePolicy-Spline ($P_{current}$: current policy, L: desired new number of parameters)

1: $K \leftarrow P_{current}$.numberOfParameters
2: $X_{current} \leftarrow [0, \frac{1}{K-1}, \frac{2}{K-1}, ..., 1]$
3: $Y_{current} \leftarrow P_{current}$.parameter values
4: $S_{current} \leftarrow$ ConstructSpline($X_{current}$, $Y_{current}$)
5: $X_{new} \leftarrow [0, \frac{1}{L-1}, \frac{2}{L-1}, ..., 1]$
6: $Y_{new} \leftarrow$ EvaluateSplineAtKnots($S_{current}$, X_{new})
7: $S_{new} \leftarrow$ ConstructSpline(X_{new}, Y_{new})
8: P_{new}.numberOfParameters $\leftarrow$ L
9: P_{new}.parameterValues $\leftarrow S_{new}.Y_{new}$
10: return P_{new}

4.2. Parameterized gaits by RL PoWER

Here we used an RL approach to change the complexity of the policy representation dynamically while the trial is running. In earlier studies on reducing energy consumption for bipedal robots (K o r m u s h e v et al. [9]), a mechanism that can evolve the policy parameterization was used. The method starts from a very simple parameterization and gradually increases its representational capability. The method was tested to be capable of generating an adaptive policy parameterization that can accommodate increasingly more complex policies. Presented in the studies of K o r m u s h e v et al. [9], the policy generated by this approach can reach the global optimum at a fast rate when applied to the energy reduction problem. Another property found about this method is its chance of converging to a suboptimal solution is reduced, because in the lower-dimensional representation this effect is less exhibited.

K o b e r and P e t e r s [8] proposed a RL algorithm named Policy learning by Weighting Exploration with the Returns (RL PoWER).

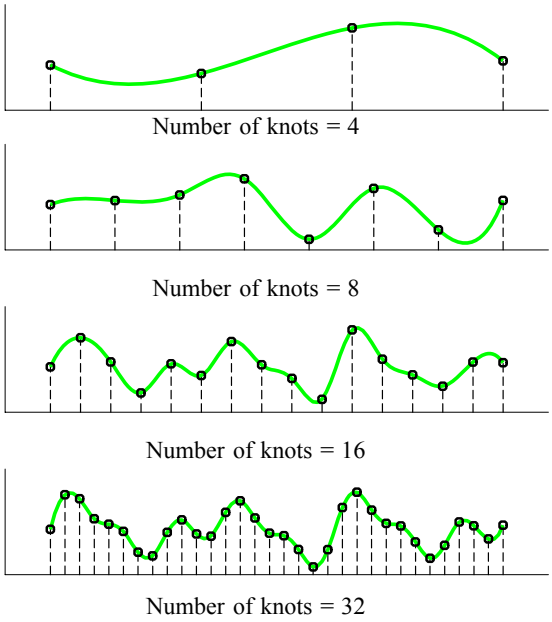


Fig. 2. An example for an evolving policy parameterization based on spline representation of the policy. The set of spline knots is the policy parameterization. The spline knots are the actual policy parameter values. This original parameterization starts from 4 knots and grows up to 32 knots

Maximization algorithm. The reason for using this is its relatively fewer parameters that need tuning. We evolved the policy parameterization only on those past trials ranked the highest by the importance sampling technique used by the PoWER algorithm. The intuition behind is that highly ranked parameterizations have more potential to evolve even better in the future. Besides, evolving all the parameterizations increases the exploring space. Since our experiment is done on a

physical robot, exploring all the variations of every parameterization is not practical. Future work may incorporate simulations into the studies, as illustrated in Bongard et al. [1].

For the experiment, we set the splines to have 3 knots for each servo, and there are 8 servos in total. The servo in the hip is not used in our experiment. Previous work has verified that quadruped gaits perform better when they are coordinated (Clune et al. [4]; Valsalam and Miikkulainen [11]). For each spline, we calculate its corresponding parameterized gait for one unit time cycle of 1.8 seconds and then apply the same pattern to every cycle throughout the 12 seconds of one trial. Specifically, each spline (a set of 3 knots) is interpreted to its corresponding servo positions as in the following equation and in Table 1.

$$g(t) = \begin{cases} R \cdot f(s1, s2, s3) + C \\ 0 + Cc \end{cases}. \quad (1)$$

Table 1. The RL PoWER motion model parameters

Parameters in θ	Description	Range
$f(s1, s2, s3)$	Spline function	[0, 1]
R	Position multiplier	[256, 768]

5. Experimental setup

The quadruped robot used, QuadraTot, was assembled from parts purchased online and parts printed by the Objet Connex 500 3D Printing System. The robot actuation system consists of 5 AX-18+ Dynamixel servos and 4 AX-12+ Dynamixel servos: one inner joint with one AX-18+ servo and one outer joint with AX-12+servo in each of the four legs, and one AX-18+ servo at the center. To avoid the formerly reported problem with AX-18+ servos are used in this robot because of their stronger actuation power than that of AX-12+. Each servo could be set to a position in the range [0, 1023] by using pydynamixel library, corresponding roughly to a physical range [–120°, +120°]. Also, to prevent collisions with the robot body, the control module filter out the commands to a safe range. This range was [–85°, +60°] for the inner leg servos, [–113°, +39°] for the outer leg servos, and [–28°, +28°] for the central hip servo. In the studies of this paper, tethered cables powered both the computer and the servos. It measures approximately 39.5 centimeters from leg to opposite leg in the crouch position shown in Fig. 1. Our performance metric was the displacement over the evaluation period of 12 seconds for each. Same as Yosinski et al. [12], the displacement was measured using a Wii remote that was placed on the ceiling. Different from the original model described in Yosinski et al. [12], the quadruped robot was equipped with a three-infrared-LED cluster on top rather than just one. The reason for this setup is that when fierce gaits were executed, the Wii remote loses tracking of the robot position due to the limited visible angle of a single LED. These three LEDs were placed tightly together to act as one signal emitter. Each LED was tilted outwards in order to maximize the visible range. A separate tracking server ran on the robot PC interacted with the Wii

remote via bluetooth by using the CWiid library. A corresponding client communicates with this server via socket connection. Our fitness evaluation code talks with the Wii remote by using this client and updates the position from beginning to end during each run. The metric for evaluating gaits was the Euclidian distance the robot travelled during a 12-second run on flat surface. Previous work done in Yosinski et al. [12] and Clune et al. [3] suggested that extremely fierce gaits are not viable in general. These gaits tend to either overburden the servos or flip the robot. Thus, RL PoWER was tested after carefully cropping out the extremely fierce gaits each time. As described earlier, each leg has two joints, inner joint $j1$ and outer $j2$. The position of one leg is determined by the sum of the values of $j1$ and $j2$, in the range of [0, 1023]. Extremely fierce gaits usually have relatively small values of $\boldsymbol{g}$. As shown in Fig. 3, this cropping method works by mapping the wild gaits right onto the boundary of a quasi-triangular area in the two-dimensional space of $j1$ and $j2$. In our experimentation, we set the threshold to 730 to balance between gait performance and safety.

Table 2. The average and standard deviation of the best gaits found for RL PoWER and HyperNEAT during each of three runs, in cm/sec. Also included is the GA gaits optimized by simulation. The data are not directly comparable because of different experiment methodology. But the best single is worth noting

Algorithm	Average	Best Single	Std. Dev
RL PoWER	**7.62**	11.05	2.1
HyperNEAT	6.28	9.7	1.26
GA	N/A	12.95	N/A

However, an obvious trade-off to this method is the reduced exploratory region of the parameters. An extension would be experiments using smaller g, cropping less gaits, to enhance the performance in general.

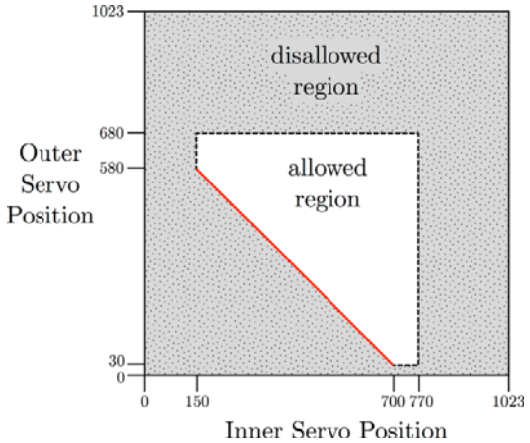


Fig. 3. The red line is the smart cropping border, $g = j1 + j2$. The dashed grey lines are the physical limits of the joints. As shown in Fig. 4, the fierce movements are prevented by mapping these out-of-range positions to the borderline as shown in the graph. The value of g cannot drop below 730, otherwise $j1$ and $j2$ are projected right to the red line

6. Results

The results for the gaits evolved by RL PoWER are shown in Fig. 5 and Table 2.The HyperNEAT performance is shown in Fig. 6. A total of 900 evaluations were performed for RL PoWER (300 for each of three runs). The reason these two algorithms have a different number of trials is that runs continued until the performance plateaued, which we defined as when there was no improvement during the last third of a run. Overall, the RL PoWER gaits were faster by far, beating the HyperNEAT when comparing either average or best gaits.

Fig. 4. The left two positions are allowed; middle two positions are on the red border line from Fig. 3; right two positions are deemed as "extremely fierce", thus are disallowed. Future work may involve studying a larger range of *g* for increased performance

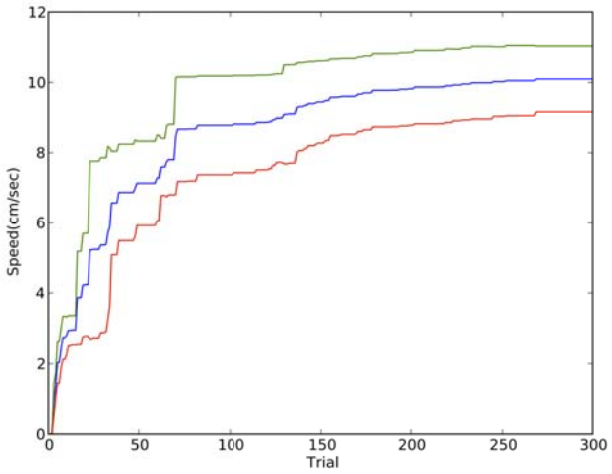


Fig. 5. Average fitness (± SE) in cm/sec of the highest performing run of RL PoWER's three runs

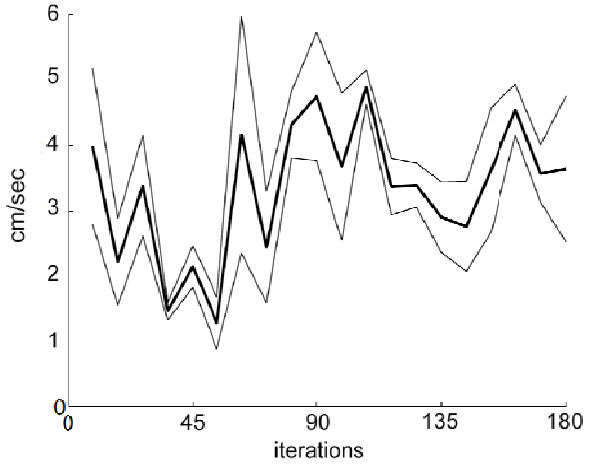


Fig. 6. Average fitness (± SE) in cm/s of the highest performing individual in one HyperNEAT run, reproduced from Y o s i n s k i et al. [12]

We believe that this is because RL PoWER uses evolvable splines to represent gaits. As explained earlier, The single best gait found during this study had a speed of 11.09 cm/s, 16.3% better than the best HyperNEAT gait. Fig. 7 shows a typical RL PoWER gait that had high fitness. Compared with Fig. 8, the pattern of RL PoWER's motion is less complex but more regular than the HyperNEAT, but producing higher speed. As shown in Fig. 7, multiple motors are more coordinated than the HyperNEAT gait.

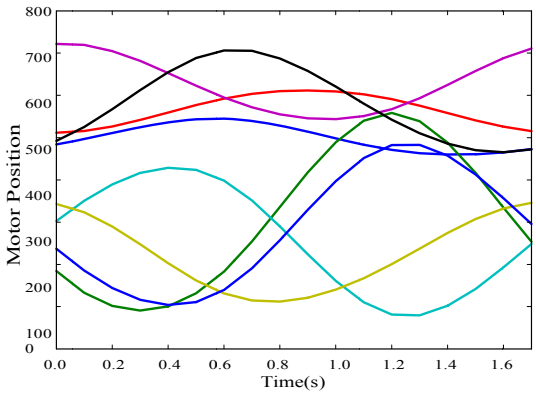


Fig. 7. The motor positions of a typical well performing gait over one cycle. According to the spline transformation, the curves of joint positions are merely linear transformations of the spline curves. Compared with the pattern of a well-performing HyperNEAT gait, the spline representation of the parameterized gaits are simple yet powerful

A corresponding observation from the graph on this property is that the noisiness of RL PoWER is higher than HyperNEAT. One reason for this is that the RL PoWER intentionally adds noise when exploring the space near the best-ranked policies. Another phenomenon seen from Fig. 5, is that one run of RL PoWER's general performance largely depends on initial gaits, which is a trait of hill climbing algorithms. The larger standard deviation in RL PoWER for space

exploring can be easily fixed by tuning the noise parameter. RL PoWER converges to optimality at a higher rate. This is due to the more heuristically guided reinforcement learning of RL PoWER. HyperNEAT, on the other hand, has a larger dimension to explore due to its generative encoding method. While the evolvable spline representation used by RL PoWER has lower dimensions. The convergence of HyperNEAT is thus slower.

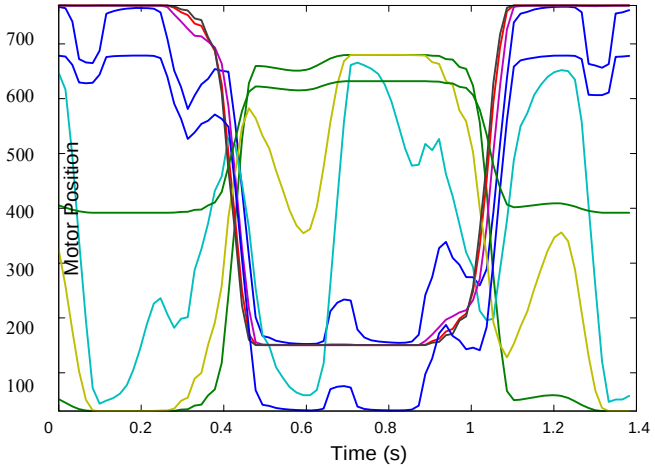


Fig. 8. A typical well performing HyperNEAT gait's pattern of motor positions, reproduced from Yosinski et al. [12]

7. Conclusions and future work

We have presented results from a recently introduced reinforcement-learning-based algorithm for optimizing a quadrupedal gait for linear speed, tested on a physical robot. We implemented the algorithm, Policy learning by Weighting Exploration with the Returns (a.k.a RL PoWER), for parameterized gaits and compared its results with those produced by HyperNEAT generative encoding and GA in a refined simulator. Each of the three methods resulted in an improvement over the robots' previous naïve gaits.

Over 900 trials have been made to investigate the applicability of RL PoWER to quadruped robots. It is difficult to gather the enough trials that would be necessary to properly rank the methods. One direction for future work could be to obtain many more trials. But due to the physical limitations, obtaining one solution to this is simulation. The results from Glette et al.'s GA algorithm, as seen in Table 2, show how simulation can help accelerate the experimentation process. Because of the low cost of simulation, it would produce the necessary volume of trials to allow the learning methods to be effective, and the hardware trials would serve to continuously ground and refine the simulator.

One hypothesis supported by this study is that for feedback-oriented tasks, reinforcement learning methods are more fit by the nature of gait learning tasks. Despite the complexities of HyperNEAT, a structurally simpler algorithm, such as RL PoWER delivered better performance in general. Also, evolvable spline interpolation

is shown to be simple and representationally powerful at the same time. Evolvable splines can serve as a general representation for various other learning problems.

Acknowledgements. This work was supported by the National Science Foundation's Office of Emerging Frontiers in Research and Innovation (Grant Number 0735953) and an NSF Postdoctoral Research Fellowship in Biology to Jeff Clune (DBI-1003220).

References

1. Bongard, J., V. Zykov, H. Lipson. Resilient Machines Through Continuous Self-Modeling. – Science, Vol. **314**, 2006, No 5802, 1118-1121.
2. Chernova, S., M. Veloso. An Evolutionary Approach to Gait Learning for Four-Legged Robots. – In: Intelligent Robots and Systems, 2004. (IROS'2004). Proceedings. 2004 IEEE/RSJ International Conference on, Vol. **3**, 2004, 2562-2567.
3. Clune, J., B. Beckmann, C. Ofria, R. Pennock. Evolving Coordinated Quadruped Gaits with the Hyperneat Generative Encoding. – In: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, 2009, 2764-2771.
4. Clune, J., K. Stanley, R. Pennock, C. Ofria. On the Performance of Indirect Encoding across the Continuum of Regularity. Evolutionary Computation. – IEEE Transactions, Vol. **15**, 2011, No 3, 346-367.
6. Glette, K., G. Klaus, J. C. Zagal, J. Torresen. Evolution of Locomotion in a Simulated Quadruped Robot and Transferral to Reality. – In: Proceedings of the 17th International Symposium on Artificial Life and Robotics, 2012.
7. Hornby, G., S. Takamura, T. Yamamoto, M. Fujita. Autonomous Evolution of Dynamic Gaits with Two Quadruped Robots. – Robotics, IEEE Transactions on, Vol. **21**, 2005, No 3, 402-410.
8. Kober, J., J. Peters. Learning Motor Primitives for Robotics. – In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, 2009, 2112-2118.
9. Kormushev, P., B. Ugurlu, S. Calinon, N. Tsagarakis, D. Caldwell. Bipedal Walking Energy Minimization by Reinforcement Learning with Evolving Policy Parameterization. – In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, 2011, 318-324.
10. Tel´lez, R., C. Angulo, D. Pardo. Evolving the Walking Behaviour of a 12 Dof Quadruped using a Distributed Neural Architecture. – Biologically Inspired Approaches to Advanced Information Technology, 2006, 5-19.
11. Valsalam, V., R. Miikkulainen. Modular Neuroevolution for Multilegged Locomotion. – In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, ACM, 2008, 265-272.
12. Yosinski, J., J. Clune, D. Hidalgo, S. Nguyen, J. C. Zagal, H. Lipson. Evolving Robot Gaits in Hardware: the Hyperneat Generative Encoding Vs. Parameter Optimization. – In: Proceedings of the 20th European Conference on Artificial Life, 2011.
13. Zykov, V., J. Bongard, H. Lipson. Evolving Dynamic Gaits on a Physical Robot. – In: Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO, Vol. **4**, 2004.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0023

Optimization of a Compact Model for the Compliant Humanoid Robot COMAN Using Reinforcement Learning

Luca Colasanto, Petar Kormushev, Nikolaos Tsagarakis, Darwin G. Caldwell

Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 1616 Genova
Emails: { luca.colasanto, petar.kormushev, nikos.tsagarakis, darwin.caldwell } @iit.it

Abstract: *COMAN is a compliant humanoid robot. The introduction of passive compliance in some of its joints affects the dynamics of the whole system. Unlike traditional stiff robots, there is a deflection of the joint angle with respect to the desired one whenever an external torque is applied. Following a bottom up approach, the dynamic equations of the joints are defined first. Then, a new model which combines the inverted pendulum approach with a three-dimensional (Cartesian) compliant model at the level of the center of mass is proposed. This compact model is based on some assumptions that reduce the complexity but at the same time affect the precision. To address this problem, additional parameters are inserted in the model equation and an optimization procedure is performed using reinforcement learning. The optimized model is experimentally validated on the COMAN robot using several ZMP-based walking gaits.*

Keywords: *Humanoid robot, Reinforcement learning, Dynamic walking.*

1. Introduction

The majority of the existing humanoid robots are powered by stiff actuation systems as in Asimo, HRP-3, iCub, LOLA and Hubo [1-11]. In fact, the predominant approach consists of using non-backdrivable, stiff transmission systems and high-gain PID controllers. This solution provides high-precision and high-load disturbance rejection but at the same time it makes the robot unsafe during interaction with humans as the environment. Moreover, the performance in

terms of energy efficiency, peak power limit and overall adaptability to the environment is very limited compared to the human being.

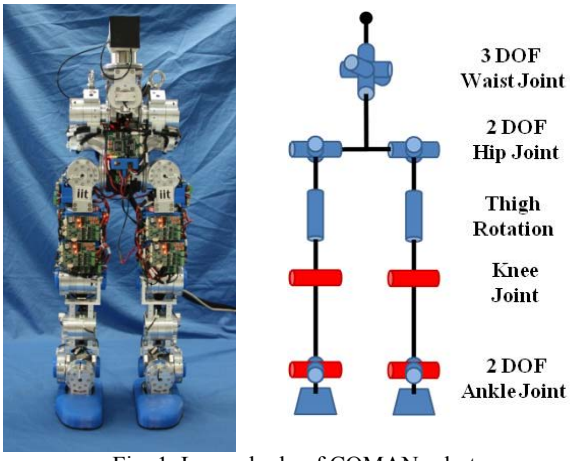


Fig. 1. Lower body of COMAN robot

Following a bioinspired approach, the new COmpliant huMANoid (COMAN) robot (Fig. 1) has been built implementing physical compliance inside the actuation system [12]. In details, passive elastic mechanism is inserted in some joints of the robot (knees and ankles) between the motor and the link. The elastic transmission gives many improvements to the robot during walking reducing the effect of foot impact with the ground. At the same time it adds extra dynamics to the system that is not presented in the common stiff robot.

In this study, a reduced model able to represent the motion of the robot including the effect of compliance is presented, then a learning technique is used to improve the performance of the model. The presentation of the work is organized as follows: In Section 2, the working principle of the model as well as the main equations of the model are presented, in Sections 3 and 4, the model parameters and the proposed optimization approach are reported.

2. Compliant humanoid model

The COMAN robot is a multi degree-of-freedom (dof), non-linear, spring-mass system because of the introduction of passive compliance in the joints. In this section, the modeling procedure to obtain a compact model of the system is reported. Following a bottom-up approach, joint dynamics is identified and then the resultant effects of all the joints are modeled using a Cartesian spring-mass-damper model at the level of the Center of Mass (CoM) [14].

2.1. Joint model

In COMAN, there are two types of joints. In the first type, the motor actuates the link through a harmonic reduction drive group. These joints are called “stiff” joints because the stiffness, due to the harmonic gearbox K_{h}, is very high. “Compliant” joints are the second type of joints where an additional physical elasticity is

incorporated in the actuation. In particular, a passive elastic mechanism is inserted in these joints between the electrical motor and the link. The additional elastic mechanism is in series with the harmonic drive and is characterized by stiffness K_s.

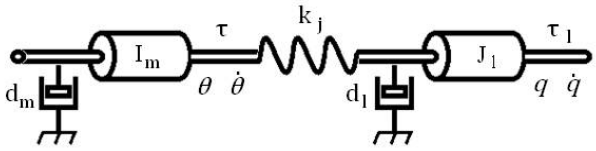


Fig. 2. Joint model

The schematic model of the joint is shown in Fig. 2. Adapting the model from [13] to this representation, the joint can be described by the following equations:

(1) $$I_\mathrm{m}\ddot{\theta} + d_\mathrm{m}\dot{\theta} = \tau - k_j(\theta - q),$$

(2) $$J_\mathrm{l}\ddot{q} + d_\mathrm{l}\dot{q} = \tau_l + k_j(\theta - q),$$

where θ, $\dot{\theta}$ and τ are the position, velocity and torque of the motor, respectively, reflected at the link side after the gear reduction:

(3) $$\theta = \frac{1}{N}\theta_\mathrm{m},$$

(4) $$\tau = N\tau_\mathrm{m},$$

where N is the gear ratio (N=100:1), θ_m and τ_m are the position and torque of the motor; I_m and d_m are the inertia and damping of the motor reflected to the link side as follows:

(5) $$I_\mathrm{m} = N^2 J_\mathrm{m},$$

(6) $$d_\mathrm{m} = N^2\left(D_\mathrm{m} + \frac{K_\tau K_\mathrm{bemf}}{R_\mathrm{m}}\right),$$

where K_τ and K_bemf are the torque sensitivity and back EMF constant, R_m is the stator resistance and D_m is the physical damping of the motor. Finally, q, $\dot{q}$, τ_l, J_l and d_l are the position, velocity, torque, inertia and damping of the link respectively and k_j is the resultant joint stiffness ($k_j = K_\mathrm{h}$ for the stiff joints and $k_j = \frac{K_\mathrm{h}K_\mathrm{s}}{K_\mathrm{h}+K_\mathrm{s}}$ in the case of compliant joints). In the case of the compliant joints, it is possible to approximate the resultant joint stiffness with K_s since $K_\mathrm{h} \approx 8000\ (\mathrm{N.m})/\mathrm{rad}$ is much larger than $K_\mathrm{s} \approx 100\ (\mathrm{N.m})/\mathrm{rad}$. From the stiffness and damping value of each joint it is possible to define the joint stiffness and damping matrix as: $K_\mathrm{j} = \mathrm{diag}(k_j^i)$ and $D_\mathrm{j} = \mathrm{diag}(d_l^i)$, i={1, 6}. Both of them are 6×6 diagonal positive definite matrices.

2.2. Cartesian model at the CoM

Based on the compliant joint model introduced in the previous section, the compliant robot behaviour is approximated by an equivalent Cartesian spring-mass-damper model at the level of the CoM.

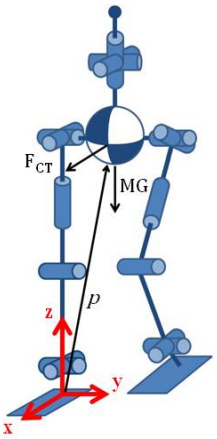


Fig. 3. Robot model and associated support foot reference frame

For each leg, the Jacobian matrix J_{CoM} from the foot base frame placed below the ankle, to the frame placed at the CoM of the robot has been computed (Fig. 3). Using the joint stiffness matrix K_{j}, the resultant Cartesian stiffness matrix $K_{\mathrm{C}} \in R^{6\times6}$ at the pelvis level (CoM) can be obtained by the following equation:

(7) $$K_{\mathrm{C}} = J_{\mathrm{CoM}}^{-\mathrm{T}} K_{\mathrm{j}} J_{\mathrm{CoM}}^{-1}$$

where $J_{\mathrm{CoM}}^{-\mathrm{T}}$ is the inverse transposed Jacobian matrix. In a similar manner, the resultant Cartesian damping matrix $D_{\mathrm{C}} \in R^{6\times6}$ at the pelvis level (CoM) can be obtained as follows:

(8) $$D_{\mathrm{C}} = J_{\mathrm{CoM}}^{-\mathrm{T}} D_{\mathrm{j}} J_{\mathrm{CoM}}^{-1}$$

where D_{j} is the joint stiffness matrix defined in the previous section. Equations (7) and (8) are an approximation of the complete relationship due to the fact that they do not take into account the change of the Jacobian matrix during the deflection movement [15]. Consequently, the approximation is valid when the deflection is small which occurs during the model experiments.

2.3. Working principle of the model

The dynamics model of COMAN is developed with the following assumptions in mind:

(A1) The joints' positions θ are controlled with a stiff PID loop.

(A2) The elasticity in the joint transmission system is due to the harmonic drive compliance as well due to additional physical elasticity integrated in the knee and ankle pitch joints of the leg.

(A3) A single mass approximation is used for the robot model.

The first assumption allows the reduction of the model's complexity. In fact, in the case of an ideally stiff position control (motor position tracking error equal to zero) and high reduction ratio (back-drivability approximately zero), the dynamics of the motor in (1) can be ignored when the robot is subject to external force

perturbations. In this case, (2) approximates the overall joint/link dynamic because the dynamics of the controlled actuator is much faster than the dynamics of the transmission.

During walking experiments, the position deflection of the CoM with respect to the desired position is large along the directions *x* and *z* and smaller along the lateral direction *y* according to foot frame (Fig. 3). This is a consequence of (A2), in fact, the level of compliance is high in the sagittal plane of the humanoid robot (due to additional elasticity in the knee and ankle pitch joints) while in lateral direction the robot is stiffer (only the compliance of the harmonic reduction drive contributes to this). Because of that, in *y* direction the movement can be approximated by a stiff system.

Finally, (A3) is an approach which has been extensively used in trajectory generation and control of humanoid robots [16]. Therefore, according to this assumption, the dynamics of the robot is approximated to the dynamics of the single mass placed at the pelvis (CoM position).

According to the previous consideration and considering equations (7) and (8), the forces generated at the pelvis (CoM) frame when the CoM position $\boldsymbol{p} = [x\ y\ z]^{\mathrm{T}} \in R^3$ deflects with respect to its reference position vector $\boldsymbol{p}_{\mathbf{r}=}[x_{\mathrm{r}}\ y_{\mathrm{r}}\ z_{\mathrm{r}}]^{\mathrm{T}} \in R^3$, can be expressed as follows:

$$F_{\mathrm{CT}} = -K_{\mathrm{CT}}(p - p_{\mathrm{r}}) - D_{\mathrm{CT}}\dot{p} \tag{9}$$

where K_{CT}, $D_{\mathrm{CT}} \in R^{3\times3}$ are sub-matrices of K_{C}, D_{C} related to the linear motion along *x*, *y* and *z*. In case of diagonal matrices *x*, *y* and *z* dynamics are completely decoupled but in this case the off-diagonal elements of the matrices in (7) and (8) are different from zero. Therefore, decoupling the movement of the robot is not possible.

The linear passive dynamics of the single-mass model can be described by the following expression:

$$M\ddot{p} = F_{\mathrm{CT}}(p, \dot{p}) + MG_{\mathrm{CT}} \tag{10}$$

where the mass-matrix $M = \mathrm{diag}(m, m, m) \in R^{3\times3}$ with m being the total mass of the robot placed at the CoM, $F_{\mathrm{CT}} = [f_{cx}\ f_{cy}\ f_{cz}]^{\mathrm{T}} \in R^3$ is the Cartesian forces given by (9) and $G_{\mathrm{CT}} = [0\ 0\ -g]^{\mathrm{T}} \in R^3$ represents the gravity.

Considering only the passive dynamics along the sagittal and vertical directions and referring to equation (9), (10) can be written in a matrix form as follows

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{z} \end{bmatrix} = -\begin{bmatrix} K_{xx} & K_{xz} \\ K_{zx} & K_{zz} \end{bmatrix}\begin{bmatrix} x - x_r \\ z - z_r \end{bmatrix} - \begin{bmatrix} D_{xx} & D_{xz} \\ D_{zx} & D_{zz} \end{bmatrix}\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ -mg \end{bmatrix} \tag{11}$$

where K_{xx}, K_{xz}, K_{zx} and K_{zz} are the relevant elements of K_{C}; D_{xx}, D_{xz}, D_{zx}, D_{zz} are the relevant elements of D_{C}; x_r and z_r are Cartesian position reference of the CoM; and x, z, $\dot{x}$ $\dot{z}$, $\ddot{x}$, $\ddot{z}$ are position, velocity and acceleration of the CoM when it is subject to external loads. The passive dynamic of the CoM of the robot in Cartesian space during stance phase is described by (11).

2.4. Reference trajectory generation

The reference trajectories used later in the learning technique and in the experimental evaluation of the model were generated based on the ZMP approach. The desired gait can be defined by a minimum set of data: step length (sl), single support duration (T_{ss}), double support duration (T_{db}). During the single support the robot was approximated with a single mass linear inverted pendulum as in [16], [17] The ZMP trajectory was computed in order to achieve the desired gait. The reference position of CoM p_{r} can be obtained from the defined ZMP reference $p_{\mathrm{ZMP}} = [x_{\mathrm{ZMP}}\ y_{\mathrm{ZMP}}\ 0]^{\mathrm{T}} \in R^3$.

$$\ddot{x_{\mathrm{r}}} - \frac{g}{Z_{\mathrm{c}}} x_{\mathrm{r}} = -\frac{g}{Z_{\mathrm{c}}} x_{\mathrm{ZMP}}, \tag{12}$$

$$\ddot{y_{\mathrm{r}}} - \frac{g}{Z_{\mathrm{c}}} y_{\mathrm{r}} = -\frac{g}{Z_{\mathrm{c}}} y_{\mathrm{ZMP}} \tag{13}$$

where Z_{c} is the fixed CoM height.

3. Model implementation

The model in (11) has been developed exploiting the characteristics of the system and adopting some approximation based on the three assumptions reported in Section 2.3. These approximations allow to reduce the complexity of the equations but at the same time they introduce an error in the model. The model equations can be rearranged as follows:

$$\xi_1 = \ddot{x} = -\alpha_1 \frac{K_{xx}}{m}(x - x_{\mathrm{r}}) - \beta_1 \frac{K_{xz}}{m}(z - z_{\mathrm{r}}) - \gamma_1 \frac{D_{xx}}{m}\dot{x} - \varepsilon_1 \frac{D_{xz}}{m}\dot{z}, \tag{14}$$

$$\xi_2 = \ddot{z} = -g - \alpha_2 \frac{K_{zx}}{m}(x - x_{\mathrm{r}}) - \beta_2 \frac{K_{zz}}{m}(z - z_{\mathrm{r}}) - \gamma_2 \frac{D_{zx}}{m}\dot{x} - \varepsilon_2 \frac{D_{zz}}{m}\dot{z}, \tag{15}$$

where α_i, β_i, γ_i and ε_i, i={1, 2} are parameters inserted to compensate model errors due to the approximation used as well as other errors from the identification of the joint stiffness and damping parameters.

Equations (14) and (15) describe the robot behavior during single support. In this phase, the robot stands on one leg therefore the robot movement is mostly affected from the compliant joints of the supporting leg. Otherwise, during double support phase, both feet are on the ground hence the compliant joints of both legs contribute to the robot movement. To take into account the effect of the second leg during this phase, the same procedure used to derive equations (9) can be reiterated for the other leg and included in the model equation. In order to reduce the complexity of the model, another approach has been adopted. Assuming that during the double support phase the two feet on the ground do not move relative to each other, the forces developed from the two legs are different because of the different configuration of the legs and different displacements. The effect of the second leg has been included to the model scaling equations (14) and (15) as follows:

$$\xi_1 = \ddot{x} = \eta_1 \left(-\alpha_1 \frac{K_{xx}}{m}(x - x_{\mathrm{r}}) - \gamma_1 \frac{D_{xx}}{m}\dot{x} - \beta_1 \frac{K_{xz}}{m}(z - z_{\mathrm{r}}) - \varepsilon_1 \frac{D_{xz}}{m}\dot{z}\right), \tag{16}$$

$$\xi_2 = \ddot{z} = -g + \eta_2 \left(-\alpha_2 \frac{K_{zx}}{m}(x - x_{\mathrm{r}}) - \gamma_2 \frac{D_{zx}}{m}\dot{x} - \beta_2 \frac{K_{zz}}{m}(z - z_{\mathrm{r}}) - \varepsilon_2 \frac{D_{zz}}{m}\dot{z} \right). \tag{17}$$

Equations (16) and (17) are used during the double support phase. The scaling coefficients η_1, η_2 are computed from experimental data which evaluates the *x* and *z* forces measured by the force/torque sensors mounted at the feet of the robot [14].

4. Optimization of the model using reinforcement learning

The proposed compact model captures well many of the characteristics of the passively compliant robot. However, it does not perfectly predict the behaviour of the robot, which is due to two main reasons: (i) the simplifying assumptions, which are necessary to keep the model compact, but also introduce approximation errors, and (ii) measurement inaccuracies when estimating the physical properties of the robot, e.g. the stiffness of the springs, etc. A common simplification assumption is, for example, that the left and the right leg have the same properties and therefore behave identically. In reality, this is not entirely true, because of the complexity of series elastic actuation, where it is normal to observe differences between the right and left leg's motors, passive compliance unit, friction, and so on.

One way to minimize the modelling error is to make the model more complex. This, however, would diminish the advantage of having a compact model, and would impede its use as a fast predictor of the robot's behaviour instead of the robot itself. Therefore, in this paper we concentrate our effort on optimizing the proposed compact model in order to achieve the best modelling precision without unnecessarily increasing the model complexity.

More concretely, by optimization of the model we mean the search for optimal values of some important parameters of the model. From equations (14)-(17) we have identified 8 important parameters whose values are crucial for the performance of the model. These parameters are as follows: α_1 and γ_1 affect the relationship between the force generate along x direction and the position and velocity of the CoM in the same direction, β_2 and ε_2 affect the relationship between the force generate along z direction and the position and velocity of the CoM in the same direction, finally β_1, ε_1, α_2 and γ_2 affect the coupling between the orizontal and vertical movement of the CoM.

Initially, the parameter values are all set to be equal to 1.0. The goal of the optimization is to find other values for these parameters which reduce the overall model error. The model error is estimated using a ground truth data set, recorded from real-world experiments with COMAN, where the reference trajectories are known, and the actual response trajectories are compared to the model output. Using this data set, we have defined a cost function to be minimized, equal to the mean squared error of the model prediction with respect to the actual response trajectories.

Many alternative optimization approaches exist, which can be used to minimize this cost function. In this paper we have selected a reinforcement learning

approach, based on direct policy search. In this approach, the policy is parameterized by a set of parameters, and the reinforcement learning algorithm is trying to optimize their values by performing a sequence of trials and evaluating its performance using the defined cost function (which is a form of reward function).

In particular, we have selected the POWER algorithm (POlicy learning by Weighting Exploration with the Returns [18]) for implementing the optimization, for the following reasons: (i) it does not require a learning rate parameter; (ii) it re-uses efficiently trials using importance sampling; (iii) it has been already applied successfully on the COMAN robot for another task, to optimize the walking gait by varying the center-of-mass height and thus reduce the energy consumption [19].

The optimization algorithm has been executed on reference trajectories with duration of 30 s (30 000 samples) and the corresponding trajectories performed by the robot computed through forward kinematics. The gait parameters are sl=0.03, T_{ss}=0.5 s and T_{db}= 0.2 s.

Table 1. Parameter values after the optimization

Parameter	α_1	β_1	γ_1	ε_1	α_2	β_2	γ_2	ε_2
Value	0.9827	1.0357	1.0560	1.1335	0.9792	0.9561	1.0072	1.1053

Table 1 contains the optimal values of the parameters found by the algorithm. The results from the optimization show that the modelling error can indeed be reduced by only changing the values of the selected eight parameters ($\alpha_1, \beta_1, \gamma_1, \varepsilon_1, \alpha_2, \beta_2, \gamma_2, \varepsilon_2$). In Fig. 4 the results of the model before and after the optimization process are compared.

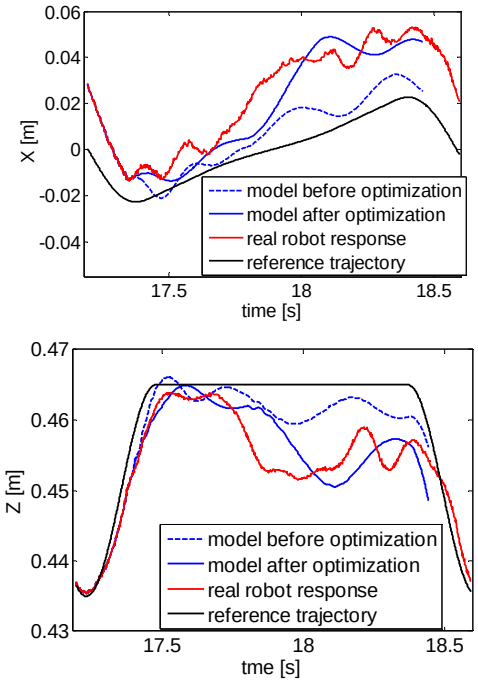


Fig. 4. Comparison of the model results before and after optimization

The base assumptions of the model reduce the precision of the inertia representation more than the gravity. The movement along the walking direction is mostly affected by the inertia of the system because there are important accelerations and decelerations. Instead vertical movement is mostly affected by gravity force. The consequence is that the movement along x directions benefit of the optimization procedure more than z direction.

5. Conclusion

In this work a reduced model of the dynamic of the CoM motion of the robot has been described. The model is based on some assumptions that reduce the complexity of the equations but at the same reduce the precision of the results. To improve the performances of the model some parameters have been inserted in the model equations in order to compensate the errors due to the reductions. The value of these parameters has been optimized using learning technique. An improvement of the model performance has been reached.

The set of parameters found has been also used to perform walking with different gaits than the one used in the optimization process. Also, in these cases the performance reached by the optimized model is better than the original model.

Acknowledgment. This work is supported by the FP7 European project AMARSI (ICT-248311).

References

1. Hirose, M., Y. Haikawa, T. Takenaka, K. Hirai. Development of Humanoid Robot ASIMO. – In: Proc. IEEE/RSJ IROS'2001, Workshop2.
2. Hirai, K., M. Hirose, Y. Haikawa, T. Takenaka. The Development of Honda Humanoid Robot. – In: Proc. of IEEE ICRA'1998, 1321-1326.
3. Akachi, K., K. Kaneko, N Kanehira, S. Ota, G. Miyamori, M. Hirata, S. Kajita, F. Kanehiro. Development of Humanoid Robot HRP-3P. – In: Proc. of IEEE-RAS Int. Conf. on Humanoid Robots, 50-55.
4. Ogura, Y., H. Aikawa, K. Shimomura, A. Morishima, H. Lim, A. Takanishi. Development of a New Humanoid Robot WABIAN-2. – In: Proc. of IEEE ICRA'2006, 76-81.
5. Tsagarakis, N. G., G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. S. Victor, A. J. Ijspeert, M. C. Carrozza, D. G. Caldwell. iCub: The Design and Realization of an Open Humanoid Platform for Cognitive and Neuroscience Research. – Advanced Robotics, Vol. **21**, 2007, No 10, 1151-1175.
6. Tsagarakis, N. G., B. Vanderborght, M. Laffranchi, D. G. Caldwell. The Mechanical Design of the New Lower Body for the Child Humanoid Robot "iCub". – In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, 4962-4968.
7. Tsagarakis, N. G., F. Becchi, M. Singlair, G. Metta, D. G. Caldwell, G. Sandini. Lower Body Realization of the Baby Humanoid-"iCub". – In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, 3616-3622.
8. Lohmeier, S., T. Buschmann, H. Ulbrich, F. Pfeiffer. Modular Joint Design for Performance Enhanced Humanoid Robot LOLA. – In: Proc. of IEEE ICRA'2006, 88-93.
9. Yamaguchi, J., E. Soga, S. Inoue, A. Takanishi. Development of a Bipedal Humanoid Robot – Control Method of Whole Body Cooperative Dynamic Biped Walking. – In: Proc. of IEEE ICRA'1999, 368-374.

10. P a r k, I. W., J. Y. K i m, J. L e e, J. H. O h. Mechanical Design of the Humanoid Robot Platform HUBO. – Journal of Advanced Robotics, Vol. **21**, 2007, No 11, 1305-1322.
11. T s a g a r a k i s, N. G., M. S i n g l a i r, F. B e c c h i, G. M e t t a, G. S a n d i n i, D. C a l d w e l l. Lower Body Design of the "iCub" a Human-Baby Like Crawling Robot. – IEEE Humanoids, 2006, 450-455.
12. T s a g a r a k i s, N., Z. L i, J. S a g l i a, D. G. C a l d w e l l. The Design of the Lower Body of the Compliant Humanoid Robot "cCub". – In: IEEE ICRA'2011.
13. T s a g a r a k i s, N., M. L a f f r a n c h i, B. V a n d e r b o r g h t, D. C a l d w e l l. A Compact Soft Actuator Unit for Small Scale Human Friendly Robots. – In: IEEE ICRA'2009, 4356-4362.
14. S p o n g, M. Modeling and Control of Elastic Joint Robots. – Trans. ASME : J. Dyn. Syst., Meas., Control, Vol. **109**, 1987, 310-319.
15. C o l a s a n t o, L., N. G. T s a g a r a k i s, D. G. C a l d w e l l. A Compact Model for the Compliant Humanoid Robot COMAN. – In: IEEE International Conference on Biomedical Robotics and Biomechatronics, Rome, Italy, 2012.
16. C h e n, S., J. K a o. Simulation of Conservative Congruence Transformation Conservative Properties in Joint and Cartesian Spaces. – In: IEEE International Conference on Intelligent Robotics and Automation, 2001, 3356-3363.
17. K a j i t a, S., M. M o r i s a w a, K. M i u r a, S. N a k a o k a, K. H a r a d a, K. K a n e k o, F. K a n e h i r o, K. Y o k o i. Biped Walking Stabilization Based on Linear Inverted Pendulum Tracking. – In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, 4489-4496.
18. K a j i t a, S., F. K a n e h i r o, K. K a n e k o, K. F u j i w a r a, K. H a r a d a H. H i r u k a v a, K. Y o k o i. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. – In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA, 2003, 1620-1626.
19. K o b e r, J., J. P e t e r s. Learning Motor Primitives for Robotics. – In: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2009, 2112-2118.
20. K o r m u s h e v, P., B. U g u r l u, S. C a l i n o n, N. G. T s a g a r a k i s, D. G. C a l d w e l l. Bipedal Walking Energy Minimization by Reinforcement Learning With Evolving Policy Parameterization. – In: Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), San Francisco, USA, September 2011, 318-324.

VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0024

Individual Recognition from Gait Using Feature Value Method

Tianxiang Zhang, Gentiane Venture

Department of Mechanical Systems Engineering, Tokyo Univ. of Agriculture and Technology
Email: venture@cc.tuat.ac.jp

Abstract: *We propose a novel framework to recognize individuals from gait, in order to improve HRI. We collected the motion data of the torso from 13 persons' gait, using 2 IMU sensors. We developed Feature Value Method which is a PCA based classifier and we achieved an average individual recognition rate of 94% through cross-validation.*

Keywords: *Gait, Recognition, PCA, Feature vector, Exclusion method.*

1. Introduction

In order to develop truly intelligent systems for Human-Robot Interaction, it is necessary to improve their ability to understand non-verbal communication. When sharing the same space as humans, robots must know whom they are interacting with. The actions to take with a stranger or a known user are different. As well within a group of users, needs and information accessibility may differ. Motion based biometrics provide a unique contact-less and non-verbal way to recognize moving individuals. It is considered as soft biometrics as it preserves privacy; and it is relatively difficult to counterfeit. It is grounded in psychological studies that have shown that one can recognize known individuals in the absence of anatomical cues by looking solely at the motion [1-7].

Techniques to record motions using motion capture are already well spread [8-9]. However, post processes based on inverse kinematics and geometrical models to calculate the joint angles from marker positions' information are necessary. In this paper we replace optical motion capture by IMU sensors. IMU sensor measures