Online Learning of Body Orientation Control on a Humanoid Robot using Finite Element Goal Babbling

Pontus Loviken[1,2,*], Nikolas Hemion[1], Alban Laflaquière[1], Michael Spranger[3] and Angelo Cangelosi[2]

***Abstract*— How can high dimensional robots learn general sets of skills from experience in the real world? Many previous approaches focus on maximizing a single utility function and require large datasets of experience to do this, something that is not possible to collect outside of simulation as every data point is expensive both in time and in a potential wear down of the robot. This paper addresses this question using a newly developed framework called *Finite Element Goal Babbling* (FEGB). FEGB is an online learning method that aims at providing general control over some measurable feature, in contrast to optimizing it to some given utility function. It generalizes standard goal babbling by breaking down the full learning problem into local sub-problems, and combining it with a planner that learns how to navigate between these sub-problems. We test FEGB using a real humanoid robot Nao, and find that it could quickly learn to robustly control its body orientation. After only 20-30 minutes of training, the robot could freely move into any body orientation between lying on either side and on its back. Rapid learning of body orientation control in high dimensional real robots is largely an unexplored field of robotics, and although many challenges remain, FEGB shows a feasible approach to the problem.**

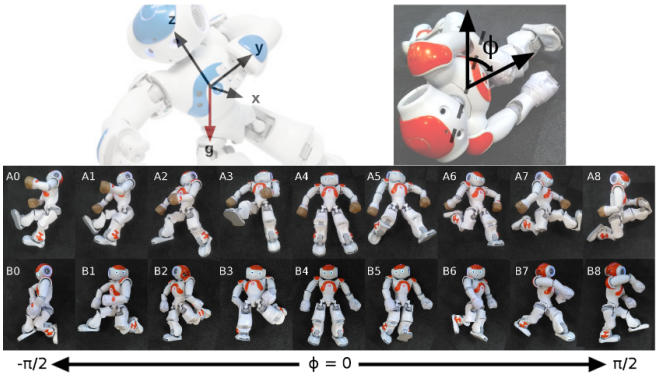

Fig. 1. Example of postures learned by FEGB, after 30 minutes of online learning. A and B represents 2 independent runs, and the number indicate the state. Each state is responsible of an interval of angle ϕ, where ϕ is the torso's orientation in relation to the ground, which can be inferred from an accelerometer inside Nao's chest that measures the direction of gravity.

I. Introduction

An open question in the field of robotics is how to allow robots to efficiently learn new behaviors or skills from experience given as little prior knowledge as possible. One way to frame this problem is within *task space* control. A task space, or *operational space*, is often a low dimensional sensor or representation space, over which getting control leads to the emergence of behaviors. "Standing up" could for example be viewed as the specific control where the 1D task space *head's-height-over-ground* is changed from a low to a high value, and "lying down" would emerge as the opposite transition. Similarly could "reaching" be described as moving the hand towards an object in the 3D task space of *hand's-position-relative-to-object*.

Separated and distinct from the task space is the *motor space*. This is the set of parameters that the robot can control directly, for example motor torque or joint angles. The challenge for the robot is to learn how to use the motor space it is in control over, in order to also get control over the task space and thus generate associated behavior(s).

In a machine learning setting, one of the most common ways to achieve such a control is through *motor babbling* [1], which consists in randomly exploring the motor space and observing the effects in the task space. This approach has many limitations however. Most striking is the problem of dimensionality. The motor space is for all but the most simple robots of very high dimension, while the outcome of a given motor command might also depend on the initial configuration of the robot. To completely explore the effect of every possible motor command from every possible starting configuration is therefore unfeasible for most real platforms. Furthermore, as illustrated in [2], such an exploration tends to sample the task space unevenly, leaving many areas of the space without any samples at all.

Goal babbling [3], [2], [4], [5] offers an alternative approach in cases where the initial configuration of the robot can be ignored. Instead of randomly exploring the motor space, this approach focuses on creating a mapping from the task space to the motor space directly. This is done by choosing goals in the task space which it tries to achieve, while simultaneously building a model for what motor command to use in order to achieve these goals. Exploring the task space instead of the motor space effectively reduces the search space to the dimensionality of the task space. Given that the task space is low-dimensional, this can lead to vastly more efficient learning.

The drawback of goal babbling is that it only applies to settings where outcomes in task space are uniquely defined by the motor command. Cases where this assumption holds includes systems that are reset to a given start configuration after every motor command (in literature referred to as a *home posture* [3]), or systems where outcomes of motor

*Corresponding author: ploviken@softbankrobotics.com
[1]AI Lab, Softbank Robotics Europe, Paris, France
[2]University of Plymouth, Plymouth, UK
[3]Sony Computer Science Laboratories Inc., Tokyo, Japan

commands are independent on the initial configuration of the system (as in [2]). This means that goal babbling can not be used in an environment where the initial configuration will change between motor commands, and where this new configuration affects the impact of a given motor command. Some attempts have been made to extend goal babbling outside of this limitation, as in [6], but so far has no framework been able to show control that can only be done by taking changing initial conditions into consideration.

Finite Element Goal Babbling (FEGB), introduced in [7], is a Finite Element Method [8] that extends goal babbling by segmenting the task space into smaller regions where each region is seen as a separate goal babbling problem. FEGB autonomously searches for a home posture to each such region so that some home posture can always be reached. It can then at a higher level learn what home postures it can transition between which allows FEGB to learn task space control even in environments where the starting configuration matters and changes with every new motor command.

In this paper FEBG is applied to the humanoid robot Nao. As task space we consider the orientation of the robot's torso relative to the ground, which is to be controlled using posture control, see Fig. 1.

So far, FEGB has only been demonstrated to work successfully in a simulation experiment. Transferring results achieved in simulation to experiments with real robots interacting with the physical world usually represents a challenge in itself. The contribution of this work is an implementation of the FEGB framework which addresses several important difficulties encountered in the real world, such as imperfect actuator control, irreversible pose transitions, and complex non-linear dynamics that change over time due to wear down, changing temperature, battery level and many other factors that are hard to predict.

The results show that FEGB allows rapid learning in this domain and the robot is able to move robustly to orientations between its back and sides after only 20-30 minutes of interaction with the environment.

Video: *http://pontusloviken.com/iros-2018*

Code: *https://github.com/loviken/fegb/tree/iros-2018*

II. Related Work

FEGB is composed of many parts and is therefore related to several different fields. This section briefly describes related work, grouped by topic.

A. Motor babbling

Motor babbling has typically been applied for models of self-perception and body map learning for manipulation. For example, Stoytchev [9] used a 2-stage process for simulating the development of self-perception with a 3-DOF robot arm. In the first stage, called motor babbling, the robot randomly moved its arm while observing its arm movements via a remote video camera. For reaching and manipulation tasks, Rucinski et al. [10] used a motor babbling strategy to allow a robot to build a spatial (left/center/right) representation of its upper body, necessary for a model of spatio-numerical association. Caligiore et al. [11] proposed a motor babbling strategy to solve a more difficult reaching problem (i.e., reaching around obstacles).

B. Goal Babbling

Goal babbling has previously been applied to a multitude of different domains, such as reaching [3], [12], [4], tool use [6], [5], and speech production [13], [14]. The motor spaces used have varied from joint angle configurations (as in [3], [2]), to motor spaces representing temporally extended motions such as Dynamic Movement Primitives (as in [6], [5]) or Central Pattern Generators (as in parts of [4]).

C. Intrinsic Motivation

Many learning scenarios contain a multitude of skills that could be learned by an agent, but only a limited amount of time in which to learn it and sample data. One approach to learn more efficiently in these cases is by rewarding actions by how much they improve the model of the agent. This is referred to in the literature as *intrinsic motivation*. Some of the most notable implementations of intrinsic motivation can be found in Intelligent Artificial Curiosity [15], where the intrinsic motivation is to optimize the learning progress in a number of different tasks, and in Empowerment [16], where the goal is to maximize the influence an agent is able to have over its surrounding. An overview of different approaches can be found in [17].

D. Hierarchical Reinforcement Learning

To some extent, the method described in this paper is related to hierarchical reinforcement learning (HRL) [18], such as the *Options* framework [19] or *Feudal Reinforcement Learning* [20]. The hierarchical structure of FEGB features a higher level planner that chooses what region to try to reach next, which in turn employs a lower level goal babbling framework in order to translate this request into a motor action. This distinguishes this hierarchy from the others in that it is a hierarchy in motor space, rather than in time, as in standard HRL.

III. Mathematical Notation

This work observes a system where an agent attempts to learn to control a low dimensional task state $x \in X \subseteq \mathbb{R}^m$ by changing its joint angle configuration $q \in Q \subseteq \mathbb{R}^n$, also referred to as a posture. It is in general assumed that the robot has many degrees of freedom, while the task space is low dimensional so that $m \ll n$. The agent can be controlled by sending a goal posture $\hat{q}$ to the agent's controller (seen as a black box) which will try to change the agent's posture q into $\hat{q}$. This attempt creates a motion until the controller stops in a new stable configuration (x', q'), where ideally $q' = \hat{q}$. In the case of goal babbling where the agent always starts from the same home configuration (x_0, q_0), or where the effect of a motor command is independent of the initial configuration, the resulting task space position x' can be controlled directly by finding an inverse mapping $g : X \to Q$ that returns a goal posture $\hat{q}$ to send to the controller in order to end up in x' in the task space [3].

IV. Finite Element Goal Babbling

This section will briefly introduce the most important concepts behind Finite Element Goal Babbling (FEGB). For more details, please refer to [7]. The fundamental intuition behind FEGB is that even though it is not possible to find a home configuration (x_0, q_0) to which the agent can always return (or reach all of the task space from), it might be possible to find a set of home configurations so that it is always possible to move to at least one of them, and then learn how to move between them.

Formally this is done by splitting the task space into a finite set of convex regions $X_s \subseteq X$ where each region is seen as the domain for a single goal babbling problem. Each such problem consists of finding a local inverse mapping $g_s : X_s \to Q$ that will work whenever moving between two configurations $(x_i, g_s(x_i))$ and $(x_j, g_s(x_j))$ for $x_i, x_j \in X_s$. In practice this means that the controller of the robot is always able to change the posture into $g_s(x_j)$ when starting from $(x_i, g_s(x_i))$, and that this change of posture will change the position in task space to x_j as well. The goal babbling problem within a region is therefore of the sort where the starting condition does not matter, as long as the configuration of the agent can be described as $(x, g_s(x))$, $x \in X_s$. If that is the case, we will say that the agent is in a state s, i.e. $(x, q) \in s \;\Leftrightarrow\; x \in X_s,\, q = g_s(x)$. Observe that it will be impossible for an agent to reach any posture $g_s(x)$ *exactly* in most real world cases, so in practice it will be necessary to have some measure of "close enough". Since a state represents a limited part X_s of task space and is also tied to the mapping g_s in motor space, every state can be thought of as a home configuration (or a home configuration space rather).

Using states it is possible to express a probability $P(s, \hat{s})$ to move between two states. It corresponds to the average probability of starting in a configuration $(x, q) \in s$, sending a motor signal $\hat{q} = g_{\hat{s}}(\hat{x})$ to the controller, and ending up in a new configuration $(x', q') \in \hat{s}$. Using these transitional probabilities it is possible for FEGB to plan transitions to any region X_{s^*} of the task space, and once there use the inverse model g_{s^*} to reach any desired position within X_{s^*}. Fig. 2 gives an overview over this process.

Given this general description of the FEGB architecture, practical design choices have to be made in order to implement it on a (real) system:

1) **Segmentation**: The task space needs to be partitioned into convex regions $\bigcup X_s = X$.
2) **Inverse model learning**: A method is needed to estimate a local inverse model g_s to every region X_s.
3) **State Recovery**: If the agent should end up in a configuration (x, q) that is not in any state s (e.g. $x \in X_s$ but $q \neq g_s(x)$), a strategy is needed to reach some state properly. This is necessary since the states acts as home postures and the dynamics of the system is only known when starting from a state s.
4) **Planner**: A strategy is needed to choose what state $\hat{s}$ to try to reach next.

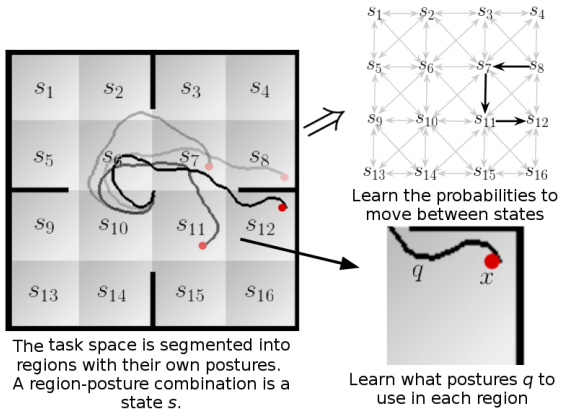

Fig. 2. This example from [7] shows how the agent is able to reach a goal on the other side of a wall by splitting the problem in two levels. One that plans a succession of states $s_8 \to s_7 \to s_{11} \to s_{12}$ to reach the region of the goal, and then an inverse model in every region that tells the agent what posture to use to reach the goal precisely.

5) **Task goal chooser**: Given a goal state $\hat{s}$ the agent needs some way to choose what task position $\hat{x} \in X_{\hat{s}}$ it should try to reach.

Once these choices are implemented FEGB can be run as indicated in Alg. 1.

Algorithm 1 Finite Element Goal Babbling (FEGB)

for iteration i **do**
  Given current state s, choose a goal state $\hat{s}$.
  Choose a task space goal $\hat{x} \in X_{\hat{s}}$.
  Send goal posture $\hat{q} = g_{\hat{s}}(\hat{x})$ to the robot's controller.
  Let the robot move until the controller converges/stabilizes.
  Observe the new configuration (x', q').
  If (x', q') is not in any state s', do *state recovery* until it is.
  Use $(x', q') \in s'$ to improve $g_{\hat{s}}(x)$, $g_{s'}(x)$ and the estimate $P(s, \hat{s})$.

V. Implementation

A. Hardware

For this study 5 different robots NAO from SoftBank Robotics were used. Nao has 25 degrees of freedom, is 58 cm tall and weights 4.3 kg. It has an accelerometer within the torso which will point in the direction of gravity at rest and can therefore be used to determine orientation ϕ, see Fig. 1. To protect the robot from damage the hands were covered with paper and tape, as can be seen in Fig. 1.

B. Adjustments due to hardware constraints

Initially the goal of the experiment was to allow control in this whole inclination space but problems were encountered as many parts of this task space could not be explored safely. Especially orientations on the belly would damage the gears of the shoulders whenever it tried to move its arms, as it is too heavy to lift itself, or whenever it fell

over after reaching a goal posture. One way to limit this damage was to decrease motor stiffness (as defined by the NAOqi API[1]), which made joints more compliant, but also weaker. This led to the problem that the agent was now no longer strong enough to learn to sit up for example, as that would require more arm strength to lift itself with. Another problem of lowered stiffness was that the controller would now undershoot posture goals. Example: A joint with value 0.0, could stop at 0.4 if asked to move to 0.5. However if it would be asked to move to 1.0 from 0.0 it would stop at 0.8. This means that the motor was in fact strong enough to reach the first goal 0.5 but would not use the necessary force when it got close enough to the goal. This is a problem since inverse models $g(x)$ are created on the assumption that if a configuration (x', q') is observed, then q' can be used as a goal posture in order to reach x', for some starting conditions. However, if the robot undershoots its goal postures this assumption falls as the robot would not return to the posture q' if asked to. As a workaround to this issue, a "wrapper" was created around the robot's controller that worked like a PI-controller [21] that would create "fake" posture targets for the robot's controller that would make it try to overshoot the true goal posture, leading to an end result closer to the real goal posture. This wrapper does not alter FEGB in any way since FEGB just perceives the controller as a black box, and it is not a problem that it is sending goal postures to this wrapper instead of to the native controller of the robot.

In the end a 1D task space was chosen corresponding to the tilt ϕ in the span between the agent's two sides, see Fig. 1. This span allowed for relatively safe exploration which was essential in order to safely develop a first FEGB implementation for the body orientation control problem.

C. Design Choices

1) Segmentation: In this experiment, the overall task space $\phi \in [-\frac{\pi}{2} - \frac{\pi}{16}, \frac{\pi}{2} + \frac{\pi}{16}]$ is divided into 9 regions X_s. Their centers are evenly distributed in order to cover the whole task space, and the width of all regions are made equal. The robot was thus limited to angles between its back and its sides.

2) Inverse model learning: The inverse model $g_s(x)$ is here approximated using Linear Regression (LR) [22], using a dataset $D_s = \{(x_i, q_i)\}_{i=1}^{N_s}$ of observations previously seen in the state. If a state has no previous observations, extrapolation can be made using the dataset of the agent's current state.

Creating a dataset D_s leads to a first problem of how to decide whether a seen configuration (x, q) should belong to the dataset or not. If no previous sample has been seen in a region it can be accepted immediately. On the contrary, if some samples have already been collected in the state, a heuristic is needed to decide if the new sample should be accepted or not. As in [7], this choice is made on the basis

[1]http://doc.aldebaran.com/2-5/naoqi/motion/control-stiffness.html

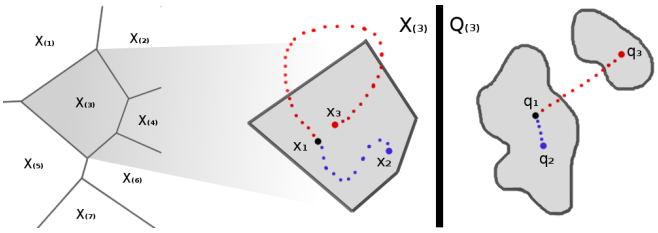

Fig. 3. An illustration of consistency. Here, $X_{(3)}$ is a region in the task space, and $Q_{(3)}$ illustrates regions in the posture space from which $X_{(3)}$ can be reached. q_1 and q_2 are here consistent with each other since it is possible to move to interpolations between them without leaving $X_{(3)}$. On the contrary, the pair q_1 and q_3 is inconsistent, since some interpolations leads to sensor states outside of $X_{(3)}$.

of *Consistency*. Consistency here refers to the ability to move to any interpolation of two postures q_1, q_2 without leaving a region X_s (see Fig. 3). An example of two non-consistent postures is sitting and standing. Both can allow the torso to point upwards, while a posture in-between most probably can not. The choice of a consistency heuristic is useful for two reasons. First, it is compatible with the choice of using LR, since averages over consistent postures will also be in the region X_s. Second, it confines the function g_s to a smaller part of the motor space, which allows the state to work as a home configuration where the agent is in a specific region of both task and motor space, see Sec. IV.

It is in practice impossible to test consistency between all seen configurations in a region. Instead, consistency is approximated using the following test: A data point (x, q) is assumed to be consistent with a state s if $x \in X_s$ and $\max\{|q - \mathrm{LR}(x|D_s)|\} < \epsilon$, where $\mathrm{LR}(x|D_s)$ is the linear regression estimate $\hat{q}$ for task space position x given dataset D_s, and ϵ is an empirically set deviation margin. This allows new samples to be added even if they differ slightly from the previous model, which allows the inverse model to adapt over time towards postures that are easier to reach, as such postures would be reached more often and thus constitute a greater proportion of the dataset. This exploration of the motor space is also important to consider when approximating the goal postures the agent will attempt to reach. Here, this is accomplished by adding a Gaussian noise to the original estimate given by LR. The exact implementation of this noise is not essential to the aim of this paper and was empirically crafted for this problem. The final expression for the inverse model is:

$$g_s(x) \approx \mathrm{LR}(x|D_s) + \left(\frac{1}{d_s(x)} + \eta\right) r\mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad (1)$$

where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a normally distributed noise-vector for exploration, $r \sim \mathcal{U}(0, 1)$ is a random scalar to modulate the over all noise rate, and $\left(\frac{1}{d_s(x)} + \eta\right)$ decreases exploration with the density $d_s(x)$ of samples around x, to allow greater exploration in the parts of task space where less samples have been observed, and η is a "background-noise" constant so that the noise level never goes completely to zero. The motivation for this background noise is that regions otherwise

risk becoming unvisitable as the robot's dynamics changed over time due to for example motor temperatures, battery levels, or wear down. In some rare cases this was however not sufficient and postures that had previously been reached became unreachable. As mappings in different states are learned independently, it is possible for a local dataset to become obsolete regarding its compatibility with neighboring states. To allow the convergence of the different datasets, we allow one to be discarded if the estimated probability to reach it falls under a threshold of 0.2. To discard a dataset ensures that a new compatible one can be found, although it is arguably sub-optimal since valuable samples might be lost. Further work therefore needs to be done to find a more optimal solution to the problem.

3) State Recovery: If an agent is in a configuration (x', q') that is not in any state s (for example by not getting close enough to its goal posture $\hat{q} = g_s(\hat{x})$, or by ending up in the wrong part of task space), state recovery will be attempted by trying to reach states randomly. First it will try to reach among the nearest states but if it fails, it will search the whole state space for recovery. Empirically this proved to be sufficient.

4) Planner: The agent is allowed to attempt transitions from its current state to its closest neighbors or itself. As the outermost states only have one neighbor, this leads to a total of 25 unique transitions that the agent can attempt, also referred to as *State-Actions*, where "Action" refers to the action of choosing where to try to go next. As in [7], this is seen as a Markov Decision Process (MDP) [23]. This MDP has a state-space $S = \{s_i\}_{i=1}^{9}$, and the action space $A_{s_i} = \{s_{i-1}, s_i, s_{i+1}\}$, where s_{i-1} and s_{i+1} are states left and right of s_i. To learn how to efficiently move around in this MDP it is necessary to approximate transitional probabilities $P(s, \hat{s})$. Observe that the underlying "true" probability to do this transition is non-stationary as the inverse models that are used to reach a state are also trained and will improve over time. How much an inverse model is improved also depends on how many times the planner has tried to reach it. This means that the planner is indirectly able to increase the probability to reach a state by attempting to go there many times. The exploration of the MDP therefore has two goals: To approximate transitional probabilities, and to identify where the inverse models can improve the most. To this end, an intrinsic reward $R(s, \hat{s})$ is used, that gives a reward every time the agent tries to move from s to $\hat{s}$ and succeeds. This reward function $R(s, \hat{s})$ is initiated to 1.0, and is then decayed by a factor $\beta \in [0, 1]$ every time the transition is successfully performed, so that $R(s, \hat{s}) \leftarrow \beta R(s, \hat{s})$. At the same time is $P(s, \hat{s})$ initiated to 1.0 and updated according to the rule $P(s, \hat{s}) \leftarrow \alpha P(s, \hat{s}) + (1 - \alpha) I(s, \hat{s})$, after every attempted transition $s \rightarrow \hat{s}$, where $\alpha \in [0, 1]$ and $I(s, \hat{s})$ is 1 if $\hat{s}$ is reached, and 0 otherwise. The overall effect of this is that a transition becomes uninteresting if it constantly fails or succeeds: If it fails because the estimated probability to receive the reward goes to zero, and if it always succeeds because the received reward goes to zero. This allows the agent to focus on the transitions where it is making the most progress and avoid transitions that are either too easy or too unlikely to succeed. Given $P(s, \hat{s})$ and $R(s, \hat{s})$ it is possible to approximate an action value function

$$Q(s, \hat{s}) = P(s, \hat{s})[R(s, \hat{s}) + \gamma V(\hat{s})] \tag{2}$$

with $V(\hat{s}) = \max_{\hat{s}'}\{Q(\hat{s}, \hat{s}')\}$ and discounted future reward $\gamma \in [0, 1]$. The planner would then choose goal states:

$$\hat{s} \leftarrow \operatorname*{argmax}_{s'}\{Q(s, s')\} \tag{3}$$

The inclusion of expected future rewards forces the agent out from the parts of task space it already masters to the parts where it can improve the most. A more detailed description of this intrinsic reward can be found in [7].

If a user would later want the robot to move to a given state s^*, this can be accomplished by changing the reward function so that only transitions to s^* are rewarded. This makes the agent plan the sequence of transitions most likely to end up in s^* (although this problem is trivial in 1D).

5) Task goal chooser: Given a goal state $\hat{s}$, a task position goal $\hat{x}$ is chosen by uniformly randomly sampling a position $\hat{x} \in X_{\hat{s}}$. The only exception is if a user has given a goal x^*, and $x^* \in X_{\hat{s}}$. Then $\hat{x} \leftarrow x^*$.

D. Hyperparameters

The hyperparameters in this experiment are determined empirically and set to:

Parameter	Value
Number of states	9
Neighborhood radius	1
Reach time	2 sec
ϵ - *Deviation margin for* D_s	$\frac{\pi}{2}$
η - *Background-noise to* $g_s(x)$	0.05
α - *Update rate for* $P(s, \hat{s})$	0.9
β - *Decay rate for* $R(s, \hat{s})$	0.7
γ - *Future reward discount for planner*	0.95

"Neighborhood radius" refers to what neighboring states the planner can try to reach, with a value of 1 being equal to only the immediately adjacent states.

VI. Evaluation

FEGB is evaluated in two ways. As it tackles a problem with no previous examples from literature a first thing to evaluate is how well it solves this problem. Secondly, although not covered in literature, it is relevant to compare it to other possible approaches.

A. Learning capability evaluation

To estimate how well FEGB learned this particular problem a natural measure is the success rate of different state-actions (SA), e.g $P(s, \hat{s})$. The planner already approximates these probabilities but, for exploratory purposes, initializes all probabilities to 1. This is too optimistic for the purpose of evaluating or visualizing the true capabilities of the agent and for that reason an additional set of probabilities $P'(s, \hat{s})$ is kept. These estimates are computed the same way as $P(s, \hat{s})$, but are pessimistically initialized to 0 and with an update rate $\alpha = 0.8$. These values are what will be used in all results.

B. Baseline comparison

Although there are no real baselines to compare the method to, it can still be possible to evaluate different parts of the solution to other approaches. In particular:

1) Does FEGB provide an efficient exploration of the task space?
2) Is consistency necessary for developing robust inverse models?

Motor babbling (MB) is a baseline for the first consideration, which is here interpreted as sampling goal postures $\hat{q}$ uniformly in the motor space of the agent. How well this strategy explores the task space can then be compared to the implementation of FEGB by observing the frequency with which different regions X_s would be visited with the respective strategies.

The second question is less straightforward to evaluate. There are many different ways inverse postures can be approximated given a number of observations $\{(x_i, q_i)\}_i$ in a given region X_s, and the importance of consistency might depend on this choice. To evaluate every possible choice of inverse model is however outside the scope of this paper, so what is evaluated is the importance of consistency for this particular set up. This is done by accepting every sample seen in a region X_s to the dataset D_s during the MB exploration. This means that the samples of D_s will be relatively evenly distributed over the motor space.

To compare the inverse models developed by the FEGB implementation to the ones by the MB approach the agent is asked to do a "random walk" through the state space by choosing neighboring states as goal states at random. Perfect inverse models would mean that the agent would spend equal amount of time in all states on average, while unreachable inverse models confines the agent to the subset of the state space it can reach. Note that it is not possible to compare success rates $P(s, \hat{s})$ here since the MB approach does not use goal states $\hat{s}$ during exploration.

C. Data collection

Training is done so that the robot always starts on its back in a relaxed posture, as in A4 or B4 in Fig. 1. If the robot falls over to its belly it is manually lifted back to its back in order to protect the shoulders. *Learning capability evaluation* is done in 10 independent runs with 5 different robots. Each robot is trained for 1000 iterations with 10 minutes rest every 200 iterations to allow the motors to cool down. Every iteration is done as described in Alg. 1. The *baseline comparison*, is done in 3 independent runs with separate robots. Every robot is here given 600 iterations with rest every 200 iterations as before. The decreased number of iterations is because MB exploration results in wild motions that are damaging to the robots. The random walk evaluation is done for 100 iterations where the agent is not allowed to improve its models with new observations.

VII. Results

Fig. 1 exemplifies two sets of postures that were found by FEGB after 1000 iterations, corresponding to around 30 minutes of training time. A video of a robot trained for 500 iterations and then requested to roll from side to side can be found at: *http://pontusloviken.com/iros-2018*

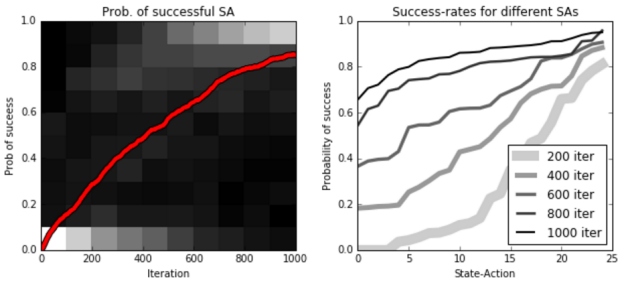

Fig. 4. Probabilities of successful state-actions (SAs), where each SA corresponds to an attempted transition between two unique states. *To the left:* The mean probability of a state-action to be successfully performed, where the background indicates the distribution over all SAs where brighter colors represents higher density. *To the right:* The mean success probabilities of every SA over 10 different runs. The SAs are here sorted by size (after averaging).

As for the quantitative evaluation, Fig. 4 shows how the success rate of SAs changes over time in FEGB. It shows that this implementation of FEGB was able to learn all SAs to this problem with a mean success-rate above 80%, with half of the SAs over 90%, after 1000 iterations.

Fig. 5 shows how often the agent would visit different regions when goal postures $\hat{q}$ were given using FEGB vs sampled uniformly from the motor space, as in the MB approach. It shows that FEGB is spending most of its time close to the middle state (on its back) to begin with, but then shifts its focus to states further and further out as it gets more proficient at moving away from the middle. This is not the case for MB where the observations are concentrated around the middlemost states, as most goal postures will move it back there. This distribution does not change over time which is expected since postures are chosen randomly.

Fig. 6 shows the distribution of time spent in different states if 100 goal states are chosen randomly (as described in Sec. VI-B). A robot that is likely to succeed with all transitions is expected to spend an equal amount of time in all states. The results shows that FEGB gets a more uniform distribution the more it is trained, meaning that it becomes more proficient at reaching each region, while the MB approach on the other hand shows a decreasing performance over time as many states that could previously be reached become unreachable. The most probable explanation for this is that MB formed non-consistent datasets when more samples were collected.

In total 5 robots were sent for repair throughout the whole process of setting this experiment up and collecting the data, but this also included trials where we wanted to see if the agent could move to positions on its belly, sitting up or while evaluating different motor stiffnesses. For this reason it is difficult to say what the wear down would typically be in the final setup. It was however clear that exploring using FEGB was much less destructive to the robots than choosing goal postures randomly as in MB. The goal postures produced by

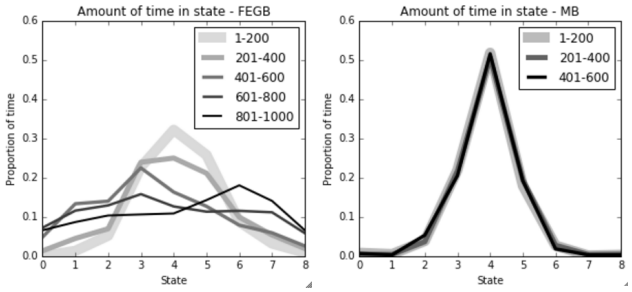

Fig. 5. Proportion of time spent in different states during training. MB is just sampling postures uniformly which explains why the distribution does not change, while the intrinsic motivation of FEGB drives it to search out the states less frequently visited, even if they might be harder to reach. FEGB is based on 10 independent runs, and MB is based on 3 independent runs.

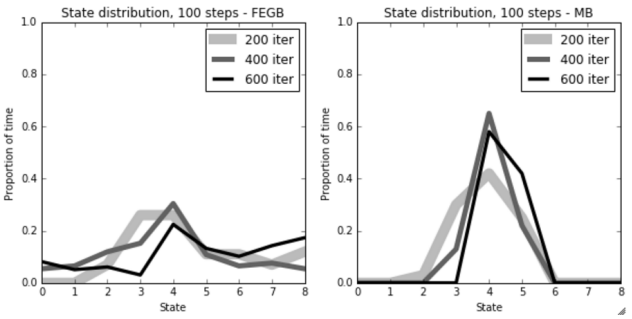

Fig. 6. Proportion of time spent in different states after 100 steps, when asked to do a random walk in the MDP and starting in state 4. The robot is not allowed to use any new observations in this test. Perfect control would approach a uniform distribution. Both charts are based on 3 independent runs.

FEGB were generally close to the robot's starting posture, and were also generally easier for the robot to reach as it was based on previously observed postures.

VIII. Discussion

This study shows a first implementation of the FEGB framework/architecture on a real high dimensional humanoid robot that rapidly learns to control its body orientation. This is a problem domain largely unexplored with no clear baseline approaches readily applicable, although it represents an important step in the sensorimotor development of infants [24]. The study shows promising results and confirms that the method is indeed well suited for the problem domain, but is so far held back by constraints such as stress on gears. It is yet not known how to get around this while still allowing high enough motor torque for the robot to effectively reach all of its inclination space. This will be particularly important if the method should be applied to robots heavier and stronger than Nao. If this problem is solved it is very likely that FEGB can also be used to learn to move between back and belly, and also to more upright postures, such as sitting up. These behaviors were seen to varying degree when setting these experiments up, but could not be systematically investigated as they could not be done safely.

Because of the focus in developing a method that allows the learning of body orientation control to begin with, rather than how to best do it, this study leaves many questions unanswered. This was to a great extent a necessary choice due to the expensive nature of the experiments, both monetarily due to wear down, and in time due to the time it takes to run experiments on physical robots. A more detailed analysis of the impact of different design choices is an important next step for future work as it would help assessing both what choices are optimal for different cases, and also what the simplest possible design choices would be for the method to work. Is it for example necessary to explore the state space using an intrinsic motivation or would random goal states work as well? Would other inverse models allow more data points to be safely added to the datasets? One way to investigate these choices more thoroughly is in simulation. This is however also at the risk of losing a lot of the effects that are essential for physical implementations, such as the problem of robot damage or changing dynamics due to factors such as wear down, changing energy levels, temperature, etc.

Another natural question from these studies is to what other problem domains it can be applied. Could the task space be extended with additional dimensions for example? Could a similar framework be used to allow a robot to learn how to stand up? There are no clear answers to these questions at the moment, but something to consider is that the learning time of the system is connected to the size of the task space, which means that adding many extra dimensions would soon lead to unfeasible training times for physical robots, assuming that the agent tries to learn to reach any position in this space. A more feasible approach would be to look at multiple task spaces in parallel, as has been done for goal babbling in for example [5] and [6]. As discussed in the introduction, many important skills can be expressed as control over low dimensional task spaces.

Acknowledgement

This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 674868 (APRIL).

References

[1] Y. Demiris and A. Dearden, "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," 2005.

[2] F. Benureau, "Self-Exploration of Sensorimotor Spaces in Robots," phdthesis, Universit de Bordeaux, May 2015. [Online]. Available: https://hal.archives-ouvertes.fr/tel-01251324/document

[3] M. Rolf, J. J. Steil, and M. Gienger, "Online goal babbling for rapid bootstrapping of inverse models in high dimensions," in *2011 IEEE International Conference on Development and Learning (ICDL)*, vol. 2. IEEE, 2011, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6037368

[4] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, Jan. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0921889012000644

[5] S. Forestier and P.-Y. Oudeyer, "Modular active curiosity-driven discovery of tool use," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 3965–3972.

[6] S. Forestier, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *arXiv preprint arXiv:1708.02190*, 2017.

[7] P. Loviken and N. Hemion, "Online-learning and planning in high dimensions with finite element goal babbling," in *7th joint international conference on development and learning and epigenetic robotics, ICDL-EpiRob 2017*, 2017.

[8] G. Dhatt, E. Lefrançois, and G. Touzot, *Finite element method*. John Wiley & Sons, 2012.

[9] A. Stoytchev, "Self-detection in robots: a method based on detecting temporal contingencies," *Robotica*, vol. 29, no. 1, p. 121, 2011.

[10] M. Rucinski, A. Cangelosi, and T. Belpaeme, "An embodied developmental robotic model of interactions between numbers and space," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, no. 33, 2011.

[11] D. Caligiore, T. Ferrauto, D. Parisi, N. Accornero, M. Capozza, and G. Baldassarre, "Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping," in *International Conference on Cognitive Systems*, 2008, pp. E1–8.

[12] M. Rolf and J. J. Steil, "Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1147–1160, Jun. 2014.

[13] C. Moulin-Frier, S. M. Nguyen, and P.-Y. Oudeyer, "Self-organization of early vocal development in infants and machines: the role of intrinsic motivation," *Frontiers in Psychology*, vol. 4, 2014. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fpsyg.2013.01006/abstract

[14] A. Philippsen, F. Reinhart, and B. Wrede, "Goal babbling of acoustic-articulatory models with adaptive exploration noise," 2016.

[15] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, Apr. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4141061

[16] C. Salge, C. Glackin, and D. Polani, "Empowerment – an Introduction," *arXiv:1310.1863 [nlin]*, Oct. 2013, arXiv: 1310.1863. [Online]. Available: http://arxiv.org/abs/1310.1863

[17] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (19902010)," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 3, pp. 230–247, 2010.

[18] B. Hengst, "Hierarchical Approaches," in *Reinforcement Learning*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 12, pp. 293–323. [Online]. Available: http://link.springer.com/10.1007/978-3-642-27645-3_9

[19] M. Stolle and D. Precup, "Learning options in reinforcement learning," in *International Symposium on abstraction, reformulation, and approximation*. Springer, 2002, pp. 212–223.

[20] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in neural information processing systems*, 1993, pp. 271–278.

[21] K. J. Åström and T. Hägglund, *PID controllers: theory, design, and tuning*. Instrument society of America Research Triangle Park, NC, 1995, vol. 2.

[22] F. Mosteller and J. W. Tukey, "Data analysis and regression: a second course in statistics." *Addison-Wesley Series in Behavioral Science: Quantitative Methods*, 1977.

[23] L. Kallenberg, "Markov decision processes," *URL http://www. math. leidenuniv. nl/˜ kallenberg/Lecture-notes-MDP. pdf*, 2009.

[24] A. Cangelosi and M. Schlesinger, *Developmental robotics: From babies to robots*. MIT Press, 2015.