Intrinsically Motivated Goal Exploration for Active Motor Learning in Robots: A Case Study

Adrien Baranes and Pierre-Yves Oudeyer
INRIA, France

Abstract— **We introduce the Self-Adaptive Goal Generation - Robust Intelligent Adaptive Curiosity (SAGG-RIAC) algorithm as an intrinsically motivated goal exploration mechanism which allows a redundant robot to efficiently and actively learn its inverse kinematics. The main idea is to push the robot to perform babbling in the goal/operational space, as opposed to motor babbling in the actuator space, by self-generating goals actively and adaptively in regions of the goal space which provide a maximal competence improvement for reaching those goals. Then, a lower level active motor learning algorithm, inspired by the SSA algorithm, is used to allow the robot to locally explore how to reach a given self-generated goal. We present simulated experiments in a 32 dimensional continuous sensorimotor space showing that 1) exploration in the goal space can be a lot faster than exploration in the actuator space for learning the inverse kinematics of a redundant robot; 2) selecting goals based on the maximal improvement heuristics is statistically significantly more efficient than selecting goals randomly.**

I. Robot Motor Learning and Constrained Exploration Mechanisms

Efficient and fast learning of forward and inverse models in high-dimensional redundant robots remains a challenge for motor learning researchers. In this paper, we introduce and evaluate an active learning approach allowing a robot to autonomously discover the inverse kinematics of its own body. This active learning approach is organized into two levels: at a higher level, the robot chooses actively goals to explore (for example points in the visual space that may be reached by its hand), and at a lower level the robot actively performs local exploration to learn how to reach goals selected at the higher level. Hence, globally the exploration is guided by motor exploration in the operational space, where goals are defined as particular configurations to reach under certain constraints (e.g. a goal may be to reach a given position with the tip of the arm through a straight line or while minimizing the spent energy). Furthermore, the selection of goals is achieved through a developmental active learning algorithm, based on work on intrinsic motivation systems [1], [2], which allows the robot to explore and learn to goals/tasks of progressively increasing complexity.

Motor learning techniques, based on machine learning techniques, are particularly useful when no robust analytical models of the kinematics and/or the dynamics of a robot exists. Yet, robots, as humans, have typically very large sensorimotor spaces, both in dimensionality and in volume relative to the length scale of the underlying forward and inverse correspondences. As a consequence, when one has no prior information on the form/structure of the underlying mathematical forward and inverse models and one is forced to use little biased statistical regression techniques, the number of training points needed for learning such models becomes extremely large. Because obtaining this training data requires physical experiments from the robot, and thus time, it becomes essential to introduce techniques that minimize the number of training examples required to reach a certain level of performance. Furthermore, it is often impossible to assume that the whole forward model can be learnt in a life-time: hence, constraints must be imposed to guide the robot such that it explores and learns only sub-parts of the model which still allow it to behave correctly in the world (and it is the same for humans).

Common ways to guide the exploration process typically set constraints on actions and goals. Social guidance is an important source of such constraints, which has been widely used in robot learning by demonstration/imitation learning [3], [4]. Here, exploration is mostly guided by an external human demonstrator. The robot programmer can also introduce constraints more directly by manually specifying specific tasks to be explored and corresponding specific reward functions. For instance, studies presented by Peters et al. [5] present a framework that combines learning by demonstration with reinforcement learning techniques: the engineer defines a specific goal in the task space and a corresponding handcrafted reward function, and a human demonstrator provides an example of a successful motor policy in the actuator space to reach that goal, which is then used to initialize an optimization procedure. Other kinds of constraints exists, in both *actuator* and *task* spaces. The Shifting Setpoint Algorithm SSA, introduced by Schaal and Atkeson in [6] also uses these two kinds of constraints. First, a goal has to be fixed by hand, and in a second time a progressive exploration process is proposed: the system explores the world gradually from the start position and towards the goal, by creating a local model around the current position, and shifting towards the goal, once this model is reliable enough; and so on.

These kinds of exploration constraints restrain the exploration to narrow tubes of data targeted at learning specific tasks/goals decided by a human (either the programmer or the demonstrator). These methods are centrally useful in many use cases. Unfortunately, in a developmental framework [7], [1], where one would like a robot to learn a variety of skills over a long period of time, we cannot conceive that a human being interacts with a robot at each instant or that an engineer designs and tunes a specific reward function for

each novel task to be learnt. For that reason, but also because generalization to unknown tasks can be needed, constrained self-exploration mechanisms must be considered.

Active learning algorithms [8] can be considered as organized and constrained self-exploration processes. These methods, generally beginning by a random and sparse exploration, are able to build meta-models of the performances of the motor learning mechanisms, and concurrently guide the exploration in various sub-spaces for which a notion of *interest* is defined, often consisting in variants of expected informational gain (e.g. heuristics for maximizing prediction errors, variance, entropy or uncertainty [9]). Yet, most of these methods assume that the space to be learnt can actually be learnt entirely in the life-time of the agent, and also assume that noise is homogeneous in that space. These assumptions, which are very useful in many applications, are again problematic in a developmental robotics framework where one is typically faced with sensorimotor spaces which are much too large to be learnt entirely, or where large subspaces may be simply unlearnable by the learning system [1], [2]. This is the reason why new active learning heuristics have been proposed, such as in the Robust-Intelligent Adaptive Curiosity (R-IAC) algorithm [2] which does not consider the current level of prediction errors as *interesting*, but rather its *local improvement* (see also [10]). Here, an interesting sensorimotor subspace is defined by the velocity of the decrease of the motor prediction errors in that subspace in a recent time window: the faster the error is decreasing, the more the expectation to obtain a still lower error is important.

Efficient when a robot is learning forward models, like consequences of its actions for given contexts, R-IAC, as other more traditional active learning algorithms have not been designed for the learning of inverse models of highly-redundant systems. Actually, they do not consider the notion of goal, or task, and only try to improve the quality of forward models with no consideration about how they can be reused for control (this applies for instance to IAC [1] and RIAC [2]). Therefore, they might typically spend large amounts of time exploring variants of actions or sequences of actions that produce the same effect, at the disadvantage of exploring other actions that might produce different outcomes and thus be useful to achieve more tasks. (e.g. learning 10 ways to push a ball forwards, instead of learning to push a ball in 10 different directions). A way to address this issue is to introduce goals explicitly and drive exploration at the level of these goals, which the system then tries to reach with a lower-level goal-reaching architecture typically based on coupled inverse and forward models, which may include a lower-level goal-directed active exploration mechanism.

In this paper, we propose an approach which allows us to transpose some of the basic ideas of R-IAC, combined with ideas from the SSA algorithm, into a muti-level active learning algorithms, called **Self-Adaptive Goal Generation R-IAC algorithm (SAGG-RIAC)**. Unlike R-IAC which was made for active learning of forward models, we show that this new algorithm allows for efficient learning of inverse models in redundant robots by leveraging the lower-level dimension of the task space. The central idea of **SAGG-RIAC**, which is to our knowledge an original approach to Competence Based Active Motor Learning as defined in [11], consists in pushing the robot to perform babbling in the goal/operational space, as opposed to motor babbling in the actuator space, by self-generating goals actively and adaptively in regions of the goal space which provide a maximal competence improvement for reaching those goals. Then, a lower level active motor learning algorithm, inspired by the SSA algorithm, is used to allow the robot to locally explore how to reach a given self-generated goal. Hence, it follows the inspiration of both the S.S.A algorithm [6], which constrains the exploration to a tube of data targeted to a specific goal, and R-IAC-like intrinsically motivated exploration, which explores in an open-ended manner the space of goals, focusing on those where local improvement of the competence to reach them is maximal.

In the following section, we introduce the Self-Adaptive Goal-Generation SAGG-RIAC algorithm as a new instantiation of the competence based intrinsic motivation framework [11]. Then, we present the SAGG-RIAC competence based active motor learning algorithm in the context of learning the inverse kinematics of an unknown simulated robot arm, and evaluate the behavior of our system as well as the quality of the learnt inverse and forward models.

II. Competence Based Intrinsic Motivation: The Self-Adaptive Goal Generation R-IAC Algorithm

A. Global Architecture

Let us consider the definition of competence based models outlined in [11], and extract from it two different levels for active learning defined at different time scales (Fig. 1):

1) The higher level of active learning (higher time scale) considers the *active self-generation and self-selection of goals*, depending on a feedback defined using the level of achievement of previously generated goals.
2) The lower level of active learning (lower time scale) considers the *goal-directed active choice and active exploration* of lower-level actions to be taken to reach the goals selected at the higher level, and depending on local measures about the evolution of the quality of learnt inverse and/or forward models.

B. Model Formalization

Let us consider a robotic system whose configurations/states are described in both an actuator space S, and an operational/task space S'. For given configurations $(s_1, s'_1) \in S \times S'$, a sequence of actions $a = \{a_1, a_2, ..., a_n\}$ allows a transition toward the new states $(s_2, s'_2) \in S \times S'$ such that $(s_1, s'_1, a) \Rightarrow (s_2, s'_2)$. For instance, in the case of a robotic manipulator, S may represent its actuator/joint space, S' the operational space corresponding to the cartesian position of its end-effector, and a may be velocity or torque commands in the joints.

In the frame of SAGG-RIAC, we are interested in the reaching of *goals*, from starting states. Also, we formalize starting states as configurations $(s_{start}, s'_{start}) \in S \times S'$ and

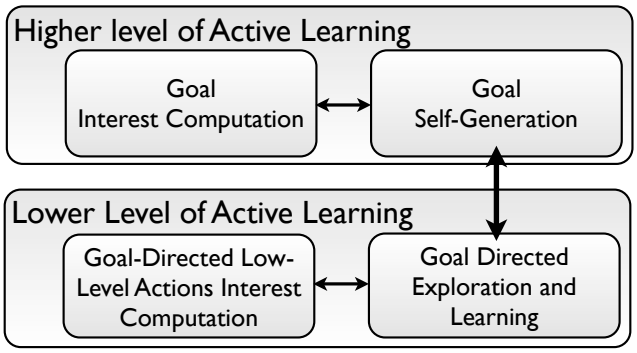

Fig. 1. Global Architecture of the SAGG-RIAC algorithm. The structure is composed of two parts defining two levels of active learning: a higher which considers the active self-generation and self-selection of goals, and a lower, which considers the goal-directed active choice and active exploration of lower-level actions, to reach the goals selected at the higher level.

goals, as a desired $s'_g \in S'$. All states are here considered as potential starting states, therefore, once a goal has been generated, the lower level of active learning always try to reach it by starting from the current state of the system.

When a given goal is set, the low-level process of goal-directed active exploration and learning to reach this goal from the starting state can be seen as exploration and learning of a motor primitive $\Pi_{(s_{start},s'_{start},s'_g,\rho,\mathbf{M})}$, parameterized by the initiation position (s_{start}, s'_{start}), the goal s'_g, constraints ρ (e.g. linked with the spent energy), and parameters of already learnt internal forward and inverse models $\mathbf{M}$.

Also, according to the self-generation and self-selection of goals at the higher level, we deduce that the whole process (higher and lower time scales) developed in SAGG-RIAC can be defined as an autonomous system that explores and learns *fields of parameterized motor primitives*.

We can easily make an analogy of this formalization with the *Semi-Markov Option* framework introduced by Sutton [12]. In the case of SAGG-RIAC, when considering an option $\langle I, \pi, \beta \rangle$, we can firstly define the initiation set $I : (S, S') \rightarrow [0;1]$, where I is true everywhere, because, as presented before, every state can here be considered as a starting state. Also, goals are related to the terminal condition β, and the policy π encodes the skill learnt through the process induced by the lower-level of active learning and shall be indexed by the goal $s_{g'}$, i.e. $\pi_{s_{g'}}$. More formally, as induced by the use of semi-markov options, we define policies and termination conditions as dependent on all events between the initiation of the option, and the current instant. This means that the policy π, and β are depending on the *history* $\{s_t, s'_t, a_t, s_{t+1}, s'_{t+1}, a_{t+1} ..., s_\tau, s'_\tau\}$ where t is the initiation time of the option, and τ, the time of the latest event. Denoting the set of all histories by Ω, the policy and termination condition become defined by $\pi : \Omega \times A \rightarrow [0;1]$ and $\beta : \Omega \rightarrow [0;1]$.

Moreover, because we have to consider cases where goals are not reachable, we need to define a *timeout* t_{max} which allows to stop a goal reaching attempt once a maximal number of actions has been executed. We thus need to consider $h_{t\tau}$, to stop π, (i.e. the low-level active learning process), if $\tau > t_{max}$.

Eventually, using the framework of options, we can define the process of goal self-generation, as the self-generation and self-selection of options, a goal reaching attempt corresponding to the learning of a particular option. Therefore, the global SAGG-RIAC process can be also described as exploring and learning *fields of options*.

C. Lower Time Scale: Active Goal Directed Exploration and Learning

The goal directed exploration and learning mechanism can be carried out in numerous ways. Its main idea is to guide the system toward the goal, by executing low-level actions, which allows to progressively explore the world and create a model that may be reused afterwards. Its conception has to respect two imperatives :

1) A model (inverse and/or forward) has to be computed during the exploration, and has to be available for a later reuse, in particular when considering other goals.
2) A learning feedback has to be added, such that the exploration is active, and the selection of new actions depends on local measures about the evolution of the quality of the learnt model.

In the experiment introduced in the following, we will use a method inspired by the SSA algorithm introduced by Schaal & Atkeson [6]. This system is organized around two alternating phases: *reaching* phases, which involve a local controller to drive the system towards the goal, and *local exploration* phases, which allows to learn the inverse model of the system in the close vicinity of the current state, and are triggered when the reliability of the local controller is too low. Other kinds of techniques, for example based on natural actor-critic architectures in model based reinforcement learning [13], could also be used.

D. Higher Time Scale: Goal Self-Generation and Self-Selection

The goal self-generation and self-selection process relies on a feedback defined using a notion of competence, and more precisely on the competence improvement in given sub-regions of the space where goals are chosen. The following part details the technical formalization of this system.

1) Measure of Competence: A reaching attempt in direction of a goal is defined as terminated according to two conditions:

- A timeout related to a maximal number of micro-actions/time steps allowed has been exceeded.
- The goal has effectively been reached.

We introduce a measure of competence for a given reaching attempt as depending on the similarity between the state s'_f reached when the goal reaching attempt has terminated and the actual goal s'_g of this reaching attempt, and the respect of constraints ρ. These conditions are measured using a cost function C, as general as the function of prediction error could be in RIAC, always negative, such that the

lower $C(s'_g, s'_f, \rho)$ will be, the more a reaching attempt will be considered as inefficient. From this definition, we set a measure of competence $\gamma_{s'_g}$ described as following:

$$\gamma_{s'_g} = \begin{cases} min_C & \text{if } C(s'_g, s'_f, \rho) \leq min_C \\ C(s'_g, s'_f, \rho) & \text{if } min_C < C(s'_g, s'_f, \rho) \leq \varepsilon_C < 0 \\ 0 & \text{otherwise} \end{cases}$$

with ε_C a tolerance factor where $C(s'_g, s'_f, \rho) > \varepsilon_C$ corresponds to a goal reached, and min_C a limit factor representing the minimal competence value. We note that a high value $\gamma_{s'_g}$ (i.e. close to 0) represents a system that is competent to reach the goal s'_g respecting constraints ρ. A typical instanciation of C, without constraints, is defined as $C(s'_g, s'_f, \emptyset) = \|s'_g - s'_f\|^2$, which is the direct transposition of prediction error in R-IAC (which here becomes goal reaching error).

2) Definition of Local Competence Progress: The active goal self-generation and self-selection relies on a feedback linked with the notion of competence introduced above, and more precisely on the monitoring of the progress of local competences. We firstly define this notion of local competence: let us consider different measures of competence $\gamma_{s'_i}$ computed for reaching attempts to different goals $s'_i \in S', i > 1$. For a subspace called a region $R \subset S'$, we can compute a measure of competence γ'' that we call a *local measure* such that:

$$\gamma'' = \left(\frac{\sum_{s'_j \in R} (\gamma_{s'_j})}{|R|} \right) \quad (1)$$

with $|R|$, cardinal of R.

Let us now consider different regions R_i of S' such that $R_i \subset S', \bigcup_i R_i = S'$. (initially, there is only one region which is then progressively and recursively split, see below). Each R_i contains attempted goals $\{s'_{t_1}, s'_{t_2}, ..., s'_{t_k}\}_{R_i}$, and corresponding competences obtained $\{\gamma_{s'_{t_1}}, \gamma_{s'_{t_2}}, ..., \gamma_{s'_{t_k}}\}_{R_i}$, indexed by their relative time order $t_1 < t_2 < ... < t_k | t_{n+1} = t_n + 1$ of experimentation inside this precise subspace R_i (t_i are not the absolute time, but integer indexes of relative order in the given subspace (region) being considered for goal selection). The interest value, described by equation 2, represents *the absolute value of the derivative of the local competence value inside R_i, hence the local competence progress, over a sliding time window of the ζ more recent goals attempted inside R_i*:

$$interest(R_i) = \frac{\left| \left(\sum_{j=|R_i|-\zeta}^{|R_i|-\frac{\zeta}{2}} \gamma_{s'_j} \right) - \left(\sum_{j=|R_i|-\frac{\zeta}{2}}^{|R_i|} \gamma_{s'_j} \right) \right|}{\zeta} \quad (2)$$

By using a derivative, the interest considers the *competence progress*, and by using an absolute value, it considers cases of *increasing and decreasing competence*. Actually, an increasing competence means that the expected competence gain in R_i is important. We deduce that, potentially, selecting new goals in subspaces of high competence progress could bring, on the one hand, a high information gain for the learnt model, and on the other hand, could lead to the reaching of not already reached goals.

Inversely, a decreasing competence in a region R_i means that some goals have been well reached in a first time, but that then, the system had been less competent to reach other goals. This means that different kinds of subregions are potentially situated in R_i (and thus R_i should be split), some where goals can be accomplished, and others, where the difficulty is too high according to the current learnt models, or which are not reachable. Therefore, the interest has also to be high for regions of decreasing competence, which explains the absolute value used here, directing the learning in these kinds of heterogeneous region, aiming to discriminate new subregions having high differences of interest value.

3) Goal Self-Generation Using the Measure of Interest: Using the previous description of interest, the goal self-generation and self-selection mechanism has to carry out two different processes:

1) Split of the space S' where goals are chosen, into subspaces, according to heuristics that allows to maximally distinguish areas according to their levels of interest;
2) Select the subspaces where future goals will be chosen;

Such a mechanism has been described in the Robust-Intelligent Adaptive Curiosity (R-IAC) algorithm introduced in [2], but was previously applied to the actuator space S rather than to the goal/task space S' as we do in SAGG-RIAC. Here, we use the same kind of methods like a recursive split of the space, each split being triggered once a maximal number of goals g_{max} has been attempted inside. Each split is performed such that is maximizes the difference of the interest measure described above, in the two resulting subspaces, this allows to easily separate areas of different interest, and thus, of different reaching difficulty.

Finally, goals are chosen according to the following heuristics which mixes three modes, and once at least two regions exist after an initial random exploration of the whole space:

1. $mode(1)$: in $p_1\%$ percent (typically $p_1 = 70\%$) of goal selections, the algorithm chooses a random goal inside a region chosen with a probability proportional to its interest value:

$$P_n = \frac{|interest_n - \mathbf{min}(interest_i)|}{\sum_{i=1}^{|R_n|} |interest_i - \mathbf{min}(interest_i)|} \quad (3)$$

Where P_n is the probability of selection of the region R_n, and $interest_i$ corresponds to the current *interest* of regions R_i.

2. $mode(2)$: in $p_2\%$ of cases (typically $p_2 = 20\%$), the algorithm selects a random goal inside the whole space.

3. $mode(3)$: in $p_3\%$ (typically $p_3 = 10\%$), it performs a random experiment inside the region where the mean competence level is the lowest.

4) Developmental Constraints for the Reduction of the Initiation Set: to improve the quality of the learnt inverse model, we add a heuristic inspired by observations of Berthier et al. [14] who noticed that infant's reaching attempts were often preceded by movements that either elevated their hand or moved their hand back to their side.

By analogy, using such heuristic can directly allow a highly-redundant robotic system to reduce the space of initiation states used to learn to reach goals, and also typically prevent it from experimenting with too complex actuator configurations. Also, we add it in SAGG-RIAC, by specifying a rest position (s_{rest}, s'_{rest}) settable without any need of planning from the system, that is set for each r subsequent reaching attempts (we call r the arm reset value, with $r > 0$).

5) Global PseudoCode: Here is the global pseudo-code of the SAGG-RIAC algorithm (Algorithm 1):

Algorithm 1 Pseudo-Code of the SAGG-RIAC Algorithm

input: **M**: empty model of the robot; ρ: constraints;
input: g_{Max} : maximal number of elements of a region
input: thresholds: ε_C; ε_{max}; *timeout*
input: rest position (s_{rest}, s'_{rest}); arm reset value: r
input: starting position (s_{start}, s'_{start})
loop
  $(s_{start}, s'_{start}) = (s_{rest}, s'_{rest})$ every r reaching attempts
  High Level of Active Learning : Part 1
  Goal Self-Generation
  Selection of a region R_n and a goal s'_g using the $mode(m)$, with probability p_m
  Reaching Attempt: Low Level of Active Learning:
  Let (s_c, s'_c) represent the current configuration of the system
  while $C(s'_g, s'_c, \rho) \leq \varepsilon_C$ & *timeout* not exceeded **do**
    Reaching Phase:
    Compute a desired $\Delta s'_i$ that minimizes $|C(s'_g, s'_c + \Delta s'_i, \rho)|$
    Perform the action a_i computed using $\mathbf{M}^{-1}$, and given the desired $\Delta s'_i$
    Get the resulting performed $\widetilde{\Delta s'_i}$ and update **M** with the element $(s_c, s'_c, a_i, \widetilde{\Delta s'_i})$
    if $\|\widetilde{\Delta s'}_i - \Delta s'_i\| > \varepsilon_{max}$ **then**
      Local Exploration Phase (e.g. as in SSA)
    end if
  end while
  High Level of Active Learning : Part 2
  Interest Update:
  Compute the competence $\gamma_{s'_g}$
  Update R_n by adding s'_g and competence $\gamma_{s'_g}$ inside
  Update the interest value $interest(R_n)$
  Split R_n if $|R_n| > g_{max}$
end loop

III. Experiment

Before discussing the details of our active exploration approach in a precise experimentation case, it is useful to consider the control paradigms involved in our study. Here, we focus on robotic systems whose actuators are settable by positions and velocities, and restrict our analysis to discrete time models.

A. Control Paradigms for Learning Inverse Kinematics

Allowing a robot to be self-adaptive to environmental conditions and changes in their own geometry is an important challenge of machine learning. These changes in the robot geometry directly have an impact on its Inverse Kinematics **IK**, relating workspace coordinates (where tasks are usually specified), to actuators coordinates (like joint position, velocity, or torque used to command the robot). Learning inverse kinematics is useful in numerous machine learning cases, like when the kinematic model of a robot cannot be accurately available or that an online calibration is needed due to sensor or motor unprecision. Also, in developmental robotics studies, the a priori of an already known precise model of the body is often avoided, because of its implausibility from the biological point of view. In the following study, we assume that the inverse kinematics of our system is totally unknown, and we are interested in studying how SAGG-RIAC can efficiently guide the discovery and learning of its inverse kinematics.

Let us mathematically formulate kinematics and inverse kinematics relations. We define the intrinsic coordinates (joint/actuator positions) of a manipulator as the n-dimensional vector $\theta \in \mathbb{R}^n$, and the position and orientation of the manipulator's end-effector as the m-dimensional vector $x \in \mathbb{R}^m$. The forward kinematic function of this system is generally written as $x = f(\theta)$, and inverse kinematics relationship is defined as $\theta = f^{-1}(x)$.

When a redundant manipulator is considered ($n > m$), or when $m = n$, solutions to the inverse relationship are generally non-unique [15]. The problem posed to inverse learning algorithms is thus to determine particular solutions to $\theta = f^{-1}(x)$, where multiple solutions exists. A typical approach used for solving this problem considers local methods, which learn relationships linking small changes $\Delta\theta$ and Δx :

$$\dot{x} = J(\theta)\dot{\theta} \tag{4}$$

where $J(\theta)$ is the Jacobian matrix $\delta f / \delta \theta$.

Then, using the Jacobian matrix and inverting it to get a single solution $\dot{\theta}$ corresponding to a desired $\dot{x}$ raises the problem of the non-convexity property of this last equation. A solution to this non-convex problem has then been proposed by Bullock in [16] who converted it into a convex problem, by only considering the learning task within the spatial vicinity $\widehat{\dot{\theta}}$ of a particular θ :

$$\dot{x} = J(\theta)\widehat{\dot{\theta}} \tag{5}$$

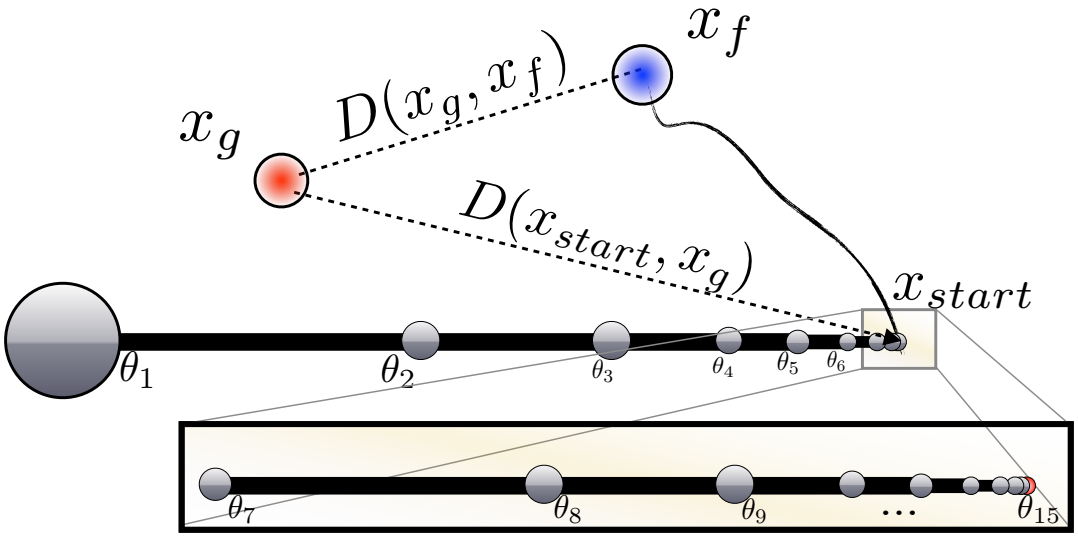

Fig. 2. Values used to compute the competence $\gamma_{s'_g}$, considering a manipulator of 15 degrees-of-freedom, in a 2 dimensions operational space. Here, the arm is set in a position called *rest position* $(\theta_{rest}, x_{rest})$.

B. Representation of Forward and Inverse Models to be Learnt

Non-parametric models typically determine local models in the vicinity of a current datapoint. By computing a model using parameterized functions on data-points restrained to a locality, they have been proposed as useful for real time queries, and incremental learning. Learning inverse kinematics typically deals with these kind of constraints, and these local methods have thus been proposed as an efficient approach to IK learning [17]. In the following study, we compute simulations and need statistical estimation, therefore, our first priority is the fast computation of queries. For this purpose, we use an incremental version of the Approximate Nearest Neighbors algorithm (ANN) [18], based on a tree split using the k-means process, to determine the vicinity of the current θ. Also, in the simulated environment that we use to introduce our contribution, we do not need highly robust, and computationally very complex regression methods. Therefore, we use the pseudo-inverse of Moore-Penrose [19] to compute the pseudo-inverse $J^+(\theta)$ of the Jacobian $J(\theta)$ in a vicinity $\widehat{\theta}$. Also, in the following equation, we use this method to deduce the change $\Delta\theta$ corresponding to a Δx, for a given joint position θ:

$$\dot{\theta} \quad = \quad J^+(\theta)\dot{x} \tag{6}$$

C. The SAGG-RIAC Algorithm Applied to a Reaching Task

In the following, we consider a n-dimensions manipulator controlled in position and speed (as many of today's robots), updated at discrete time values, called *time steps*. The vector $\theta \in \mathbb{R}^n = S$ represents joint angles, and $x \in \mathbb{R}^m = S'$, the position of the manipulator's end-effector in m dimensions, in the euclidian space S' (see Fig. 2 where $n = 3$ and $m = 2$). We evaluate how the SAGG-RIAC algorithm can be used by a robot to learn how to reach all reachable points in the environment S' with this arm's end-effector. Learning the inverse kinematics is here an online process that arises each time a micro-action is executed by the manipulator: by doing movements, the robot stores measures $(\theta, \Delta\theta, \Delta x)$ in its memory; these measures are then reused online to compute the Jacobian $J(\theta) = \Delta x / \Delta\theta$ locally to move the end-effector in a desired direction $\Delta x_{desired}$ fixed towards the self-generated goal. Therefore, we consider a learning problem of $2n$ dimensions, the relationship that the system has to learn being $(\theta, \Delta\theta) \Rightarrow \Delta x$. Also, in this experiment, where we suppose S' euclidian, and do not consider obstacles, the direction to a goal can be defined as following a straight line between the current end-effector's position and the goal.

1) Evaluation of Competence: In this experiment, we do not consider constraints ρ and only focus on the reaching of goal positions x_g. We define the cost function C and thus the competence as linked with the euclidian distance $D(x_g, x_f)$, between the goal position and the final reached position x_f, which is normalized by the starting distance $D(x_{start}, x_g)$, where x_{start} is the end-effector's starting position:

$$C(x_g, x_f, x_{start}) \quad = \quad -\frac{D(x_g, x_f)}{D(x_{start}, x_g)} \tag{7}$$

where $C(x_g, x_f, x_{start}) = min_C$ if $D(x_{start}, x_g) = 0$ and $D(x_g, x_f) \neq 0$.

2) Addition of subgoals: Computing local competence progress in subspaces/regions typically requires the reaching of numerous goals. Because reaching a goal can necessitate several actions, and thus time, obtaining competence measures can be long. Also, without biasing the learning process, we improve this mechanism by taking advantage of the euclidian aspect of S': we increase the number of

goals artificially, by adding subgoals on the pathway between the starting position and the goal, where competences are computed. Therefore, considering a starting state x_{start} in S', and a self-generated goal x_g, we define the set of l subgoals $\{x_1, x_2, ..., x_l\}$ where $x_i = (i/l) \times (x_g - x_{start})$, that have to be reached before attempting to reach the terminal goal x_g.

D. Local Exploration and Reaching

In [6], Schaal & Atkeson propose a method called SSA to deal with learning sparse data in high dimensional spaces. Based on the observation that random exploration could be very long, unsafe, or costly, they introduced an exploration algorithm decomposing the problem of motor control, into two separated control tasks: a first one, where it trains a nonlinear regulator, by directing the controlled system to stay close to some chosen setpoints. And a second one where setpoints are shifted to reach a handcrafted goal. This typically allows the system to create a narrow tube of known data, to guide it toward the goal.

Here we propose a method, inspired by the SSA algorithm, to guide the system to learn on the pathway toward the selected goal position x_g. The system is organized around two alternating phases: *reaching* phases, which involve a local controller to drive the system from the current position x_c towards the goal, and *local exploration* phases, which allows to learn the inverse model of the system in the close vicinity of the current state, and are triggered when the reliability of the local controller is too low. These mechanisms are stopped once the goal has been reached or a timeout exceeded. Let us here describe the precise functioning of those phases in our experiment:

1) Reaching Phase: the reaching phase deals with creating a pathway to the goal position x_g. This phase consists of determining, from the current position x_c, an optimal movement to guide the end-effector toward x_g. For this purpose, the system computes the needed end-effector's displacement $\Delta x_{next} = v.\frac{x_c - x_g}{\|x_c - x_g\|}$ (where v is the velocity bounded by v_{max} and $\frac{x_c - x_g}{\|x_c - x_g\|}$ a normalized vector in direction of the goal), and performs the action $\Delta\theta_{next} = J^+.\Delta x_{next}$, with J^+, pseudo-inverse of the Jacobian estimated in the close vicinity of θ and given the data collected by the robot so far. After each action Δx_{next}, we compute the error $\varepsilon = \|\widetilde{\Delta x}_{next} - \Delta x_{next}\|$, and trigger the exploration phase in cases of a too high value $\varepsilon > \varepsilon_{max} > 0$.

2) Exploration Phase: this phase consists in performing $q \in \mathbb{N}$ small random explorative actions $\Delta\theta_i$, around the current position θ. This allows the learning system to learn the relationship $(\theta, \Delta\theta) \Rightarrow \Delta x$, in the close vicinity of θ, which is needed to compute the inverse kinematics model around θ.

IV. Results

A. Goal Directed Exploration and Learning

In the experiment introduced in this section, we consider the robotic system presented above with a n=15 DOF arm on a plane (thus the problem has 32 continuous dimensions, with 30 dimensions in the actuator/state space and 2 dimensions in the goal/task space) . We set the dimensions of S' as bounded in intervals $x_g \in [0;150] \times [-150;150]$, where 50 units is the total length of the arm, which means that the arm covers less than $1/18$ of the space S' where goals can be chosen (i.e. the majority of areas in the operational/task space are not reachable, which has to be discovered by the robot). We fix the number of subgoal per goal to 5, and the maximal number of elements inside a region before a split to 50. We also set the desired velocity $v = 0.5$ units/movement, and the number of explorative actions $q = 20$. Moreover, we reset the arm to the rest position $(\theta_{rest}, x_{rest})$, where the arm is straight (position displayed in Fig. 2), every $r = 2$ reaching attempts. This allows to reduce the initiation set, and prevent the system from experimenting with too complex joint positions where the arm is folded, and where the jacobian is more difficult to compute.

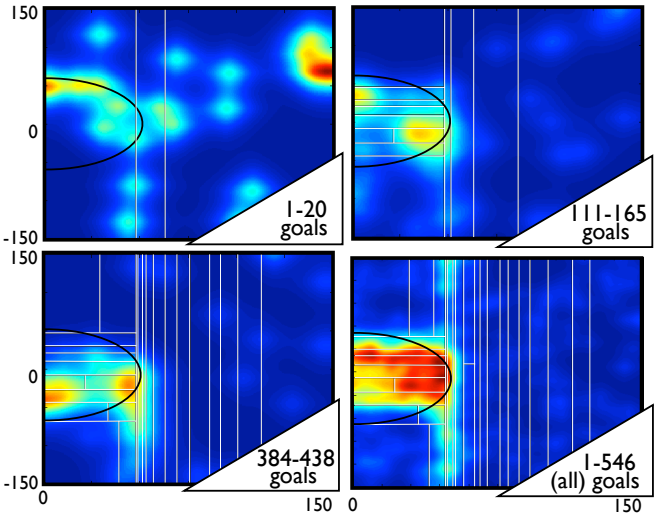

Fig. 3. Histograms of self-generated goals and regions (split by white lines) displayed over time windows indexed by the number of performed goals, for an experiment of 200000 time steps (i.e. micro-actions). The black half-circle represents the contour of the area reachable by the arm according to its length of 50 units

Fig. 3 shows histograms of the self-generated goal positions (goals without subgoals), and created regions, after the execution of 200000 time steps (i.e. micro-actions). Each subfigure represents data obtained during a time window indexed on the number of generated goals: the first one (upper-left) shows that in the very beginning of learning (20 goals corresponds to 100 goals+subgoals), the system is already splitting the space and seems to discriminate the left third of the space, where the reachable area is (contoured by the black half-circle on each subfigure). Upper-right and lower-left subfigures show examples of areas where goals are generated over time, we can observe that the highest amount of goals that are chosen remains inside the reachable area: the system is indeed discovering that only a subpart is reachable, the interest value becoming null in totally unreachable areas where the competence typically takes small values, or even reach the threshold min_C. The last subfigure (lower-right)

represents the position of all goals that have been self-generated and allows to observe that SAGG-RIAC is able to highly discriminate unreachable areas over time, and to focus its goal self-generation in the whole reachable subspace. Finally, observing regions, we can globally notice that the system split the reachable space into regions in the first quarter of goal generations (upper-right subfigure), and then continue to split the space inside unreachable regions, in order to potentially release new areas of interest.

It is also important to notice that coupling the lower-level of active learning inspired by SSA with the heuristic of returning to x_{rest} every two subsequent goals creates an increasing radius of known data around x_{rest}, inside the reachable space. Indeed, the necessity to be confident in the local model of the arm to shift toward new positions makes the system progressively explore the space, and resetting it to its rest position makes it progressively explore the space by beginning close to x_{rest}. Finally, goal positions that are physically reachable but far from this radius typically present a low competence to be reached initially, before the radius spreads enough to reach them, which creates new areas of interest, and explains the focalization on reachable areas far from x_{rest} (see Fig. 3). Therefore, the exploration also proceeds by going through reachable subspaces of growing complexity of reachability.

B. Quantitative Results

In the following evaluation, we consider the same robotic system as previously and design two experiments.

1) Experiment using $S' = [0;50] \times [-50;50]$: The first experiment is designed to compare the quality of forward and inverse models using babbling in the task/operational space, i.e. using goals, instead of more traditional motor babbling heuristics which are executed in the configuration/actuator space. Therefore, we still consider a n=15 DOF arm of 50 units, and, to be suited for this study, the dimensions of S' are bounded in intervals $x_g \in [0;50] \times [-50;50]$ which means that the arm can reach almost all the surface S' where goals can be chosen. Here we observe the system during 50000 movements, and fix $q = 20$.

We compare 4 different types of exploration techniques:

1) SAGG-RIAC
2) SAGG-Random, where goals are chosen randomly (higher-level of active learning (RIAC) disabled)
3) ACTUATOR-Random, where small random movements $\Delta\theta$ are executed.
4) ACTUATOR-RIAC, which corresponds to the original RIAC algorithm, which uses the decrease of the prediction error $(\theta, \Delta\theta) \rightarrow \Delta x$ to compute an interest value and split the space $(\theta, \Delta\theta)$.

Also, to be comparable to SAGG-RIAC, for each other techniques we reset the position of the arm to the *rest position* every *max* time steps, *max* being the number of time steps needed to consecutively reach the two more distant reachable positions. We present statistical results obtained after launching this same experiment with different random seeds 15 times.

Fig. 4 (left) shows the evolution of the capability of the system to reach 100 test goals (independantly and uniformly distributed in the reachable area) using the inverse model learnt by each technique, starting from, half the time, the rest positions. This capability is computed using the euclidian distance between the goal and the final state of a reaching attempt.

Fig. 4 (right) illustrates the quality of forward models learnt at the same time, tested using the prediction error of the end-effector positions, according to a test database of 100 joint configurations.

Globally, these results show that for learning the inverse kinematics of this highly-redundant arm, exploration in the goal/operational space is significantly more efficient than exploration in the actuator space using either random exploration or RIAC-like active learning. Moreover, comparison of ACTUATOR-Random and ACTUATOR-RIAC emphasize that the original version of RIAC has not been designed for an efficient learning of forward and inverse models of highly-redundant systems, which explains its efficiency lower than ACTUATOR-Random in both graphs of Fig 4.

2) Experiment using $S' = [0;150] \times [-150;150]$: This second experiment considers a larger space $S' = [0;150] \times [-150;150]$, where it is more relevant to compare SAGG-RIAC and SAGG-Random because of the capability of SAGG-RIAC to discriminate unreachable areas. Here, we fix $q = 100$, which allows a precise exploration of localities when the model has been judged as not efficient by the reaching phase of the low-level of active learning, and compute tests of inverse and forward models over 200000 time steps (i.e. micro-actions).

Fig. 5 illustrates the efficiency of inverse and forward models over time (number of *time steps*). We can firstly observe the lower decreasing velocity of SAGG-Random and SAGG-RIAC, compared to the previous experiment, which is due to both the expansion of S', and the higher value of q. Observations of ACTUATOR-RIAC and ACTUATOR-Random shows the same kind of behavior than before, the reaching and prediction errors being almost stabilized after a first decrease before 20000 movements. Moreover, the main observation that can be extracted from these graphs is the continuous evolution of mean reaching and prediction errors of SAGG-RIAC under errors values of SAGG-Random since 20000 movements; also, ANOVA analysis on the reaching error of SAGG-RIAC and SAGG-Random shows a level of significance $p = 0.002$, and $p = 0.0065$ on the prediction error, at the end of the experiment (200000 time steps). Therefore, the proposed mixture of SAGG and RIAC algorithms leads to significant better performances, according to the learning of both inverse and forward kinematics of this kind of highly-redundant system.

More systematic studies should be done on these kind of comparisons, but, these results indicate the high potential of **competence based motor learning** in general, even using random Goal Self-Generation, for IK learning in redundant robots.

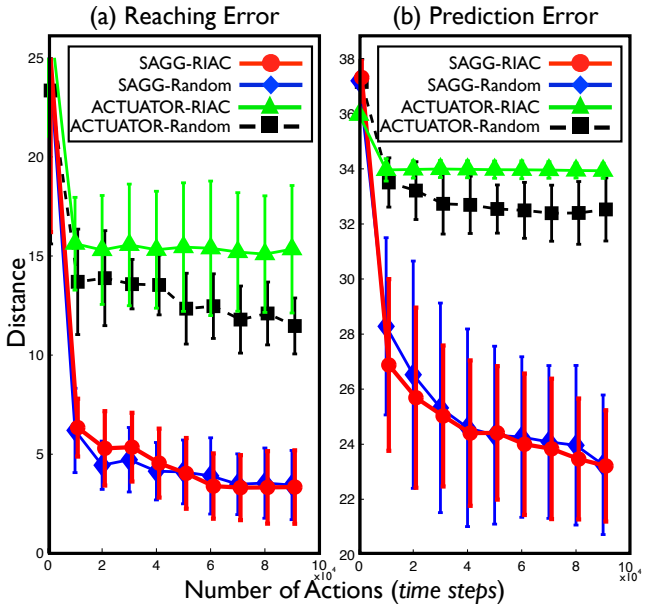

Fig. 4. Evolution of mean distances goal-end effector after reaching attempts over an independantly randomly generated set of test goals (reaching error (a)), and prediction error on an independantly and randomly generated set of test configurations (prediction error (b)). Here SAGG-RIAC and SAGG-random are only allowed to choose goals within $S' = [0;50] \times [-50;50]$ (i.e. most eligible goals are physically reachable). Standard deviations are computed at the same instants for each curve, and shifted in graphs for an easy reading.

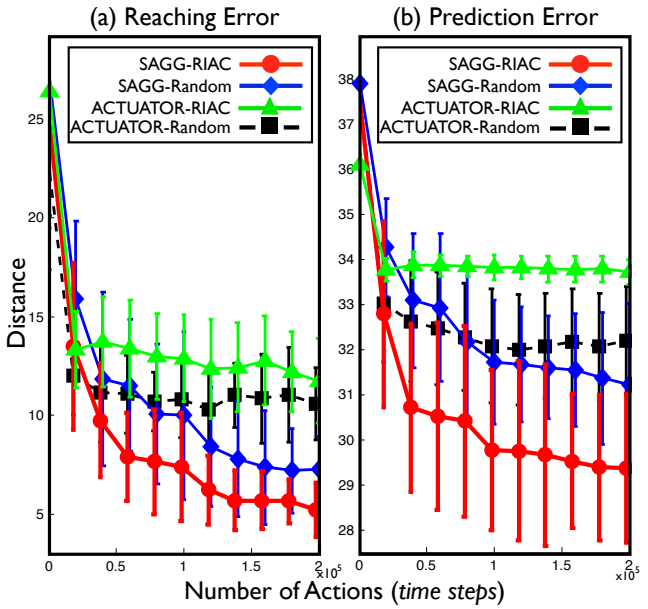

Fig. 5. Evolution of mean distances goal-end effector after reaching attempts over an independantly randomly generated set of test goals (reaching error (a)), and prediction error on an independantly and randomly generated set of test configurations (prediction error (b)). Here SAGG-RIAC and SAGG-random are only allowed to choose goals within $S' = [0;150] \times [-150;150]$ (i.e. the set of reachable goals is only a small subset of eligible goals).

V. Conclusion

This paper presented the Self-Adaptive Goal Generation algorithm, SAGG-RIAC, in a Competence Based Active Motor Learning framework. We showed that SAGG-RIAC can be a very efficient exploration mechanism to learn the inverse kinematics of high-dimensional redundant robots, by actively guiding them both for high-level self-generation of goals in the task space and for low-level exploration while learning to reach these chosen individual goals. An important future direction will consist in transposing the experiments we presented in real robotic setups as well as in various other sensorimotor embeddings to evaluate the scalability of the results presented in this paper.

VI. Acknowledgment

This research was partially funded by ERC Grant EXPLORERS 240007

References

[1] P.-Y. Oudeyer, F. Kaplan, and V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11(2), pp. pp. 265–286, 2007.

[2] A. Baranes and P.-Y. Oudeyer, "Riac: Robust intrinsically motivated exploration and active learning," *IEEE Transation on Autonomous Mental Development*, vol. 1, no. 3, pp. 155–169, December 2009.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Handbook of Robotics*. Springer, 2008, ch. Robot Programming by Demonstration, pp. 1371–1394.

[4] S. Calinon and A. Billard, "Statistial learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, no. 15, pp. 2059–2076, 2009.

[5] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Humanoids'2003, Third IEEE-RAS International Conference on Humanoid Robots*, 2003.

[6] S. Schaal and C. G. Atkeson, "Robot juggling: an implementation of memory-based learning," *Control systems magazine*, pp. 57–71, 1994.

[7] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, no. 599-600, 2001.

[8] V. Fedorov, *Theory of Optimal Experiment*. New York, NY: Academic Press, Inc., 1972.

[9] S. Thrun, "Exploration in active learning," in *Handbook of Brain Science and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995.

[10] J. Schmidhuber, "Curious model-building control systems," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2, 1991, pp. 1458–1463.

[11] P.-Y. Oudeyer and F. Kaplan, "How can we define intrinsic motivations ?" in *Proc. Of the 8th Conf. On Epigenetic Robotics.*, 2008.

[12] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 1123, no. 181-211, 1999.

[13] J. Peters and S. Schaal, "Natural actor critic," *Neurocomputing*, no. 7-9, pp. 1180–1190, 2008. [Online]. Available: http://www-clmc.usc.edu/publications//P/peters-NC2008.pdf

[14] N. E. Berthier, R. Clifton, D. McCall, and D. Robin, "Proximodistal structure of early reaching in human infants," *Exp Brain Res*, 1999.

[15] B. Siciliano and O. Khatib, *Handbook of Robotics*. Springer, 2008.

[16] D. Bullock, S. Grossberg, and F. Guenther, "A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm," *Journal of Cognitive Neuroscience*, vol. 5, no. 4, pp. 408–435, 1993.

[17] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[18] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm," in *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, 2009.

[19] A. Albert, "Regression and the moore-penrose pseudo inverse," in *Mathematics in science and engineering*. Academic Press, Inc., 1972.