2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)
November 3 - 5, 2015, Seoul, Korea

Where is my keyboard? Model-based active adaptation of action-space in a humanoid robot

Arturo Ribes, Jesus Cerquides, Yiannis Demiris, Ramon Lopez de Mantaras

***Abstract*— Nowadays robots are becoming more ubiquitous, and focus is increasingly put on the variety of tasks they can perform autonomously. However, due to the dynamics of the environment or the robot itself, sometimes the models that were learned in the past do not exactly fit in the present situation. Adaptation mechanisms are of key importance in this case since they enable the robot to reuse prior knowledge, which is useful for the current context. In this work we introduce an active adaptation mechanism which enables a humanoid robot to recover from a failure, exemplified as a displacement of the object it is interacting with.**

I. Introduction

In order for robots to interact with their environment, they must choose actions based on a given set of goals, which may be internally or externally generated. That is, the robot may be generating its own goals or these can be provided by a human supervisor.

Usually the robot will have an internal model which maps the task space or goal space into an internal action space. This task space should be some measure that the robot can obtain though its sensors. This follows from the *verification principle*, which basically states that successful AI systems should be able to verify their knowledge, or in this case their correct interaction with their environment [1]. This principle, along with others cited in the work of Stoytchev [1], provides the robot with a good degree of autonomy.

In order to support life-long learning, a robot must also be able to adapt to changes in its environment or even in its own body. As these changes will ultimately affect the robot performance, the robot should reuse its knowledge in order to update it to reflect the changes, effectively restoring its previous performance level.

We consider this ability very important to improve the degree of autonomy of a robot, specially in the context of humanoid robots, which in the long run will have a lot of learned skills and if for some reason one is affected by some change, the robot should not need to relearn it but to adapt to the change.

As an example, imagine a blind pianist robot that learned how to use a virtual keyboard like the one shown in Figure 1. After the model is learned, the robot knows the spatial distribution of the keys. But if the keyboard position is changed, the robot should not relearn the new distribution of keys, but rather adapt the old to the new one. Taking advantage of the already learned knowledge will speed up this process.

Fig. 1. iCub interacting with the virtual keyboard shown by the Reactable tactile interface. The finger is used to control the virtual object, which is used by our software to know which sound to play.

In this work, we present a method for active adaptation of the robot action space, in order to find the displacement undergone by the virtual keyboard after its location was perturbed. Our contribution is two-fold. One one hand, we propose an alternative way to represent the hypotheses space, that is, the distribution of possible displacements obtained from a sequence of observations made by the robot after the perturbation occurred. This is done by encoding this posterior distribution as a product of kernel density estimates. On the other hand, we introduce an active learning strategy in order to select actions which are expected to reduce the number of observations needed to recover the correct displacement. In this way, the robot can restore its previous performance in a quicker way.

II. Related work

The topic of adaptation of a learned model to a new situation can be seen in various areas related to robotics. In the area of global localization, the task is to find the location of the robot in a map by using information from the robot sensors and a previously learned model. This model contains the locations where certain features can be found in a map. The features can be from laser or sonar sensor readings [2][3]

Arturo Ribes is with the Learning Systems department at IIIA-CSIC (UAB), Campus de la UAB, E-08193, Bellaterra, Barcelona (Spain) and the Personal Robotics Lab at the Imperial College of London, London, UK. e-mail: aribes@iiia.csic.es.

Jesus Cerquides is at IIIA-CSIC (UAB), Campus de la UAB, Bellaterra, Barcelona (Spain).

Yiannis Demiris is with the Personal Robotics Lab at the Imperial College of London, London, UK.

Ramon Lopez de Mantaras is at IIIA-CSIC (UAB), Campus de la UAB, Bellaterra, Barcelona, Spain.

978-1-4799-6885-5/15/$31.00 ©2015 IEEE

to visual features like image patches [4][5] or even the output of object detectors for doors and windows [6].

Most approaches nowadays are based on the Monte-Carlo Localization (MCL) formulation introduced in [2]. The main idea is to decrease the uncertainty in the random variable representing the location of the robot by a successive estimation of its posterior under a Markov assumption. A major difference between the robot localization problem and our problem resides in the fact that in the localization problem, the actions of the robot change its location, which is what we are trying to estimate. In our problem, the object is displaced to a new position and remains there. However, the actions of the robot do not affect the object location. Despite that fact, the incremental estimation of the posterior over the location random variable and the active techniques introduced are similar to ours.

The most important issue that is related to our work is how the model encodes the relationship between features and locations in the map. The kind of features which are best suited for localization are point features, that is, features that occur in specific location in the map [4][6][7][8]. In this way, an observation of a feature is transformed into a single hypothesis, or a set of potential candidate hypotheses, with a clear location and an estimate of the associated uncertainty, which will be refined with successive estimates.
However, given the nature of our problem, we do not have point features, but rather area features, that is, the sensor readings encoded in our model are distributed over an area of an arbitrary shape, rather than a point with an uncertainty belonging to a known distribution family. This makes the use of Kalman filters particularly problematic, so alternative methods make use of particle filters to represent such distributions [5][8].

When having to select among different candidate hypotheses, it is interesting to keep track of multiple hypotheses [6][3]. In [6], authors represent each hypothesis with a Kalman filter and then assign each new observation to the hypothesis which provides a better support.
Instead of representing the hypotheses space with a few candidates, we use a kernel density estimate of the hypotheses distribution and extract the best hypothesis when needed.

The other aspect that we study here is the active choice of actions in order to reduce the number of observations needed to solve the problem. Murtra et al. [3] provide an interesting separation of active strategies for global localization, namely heuristic based, geometry based and information theoretic based. While some methods report good results using a heuristic based active strategy [6][7], given that we already have a probabilistic representation of both the model and the hypotheses space, we can take advantage of information theoretic strategies.

There are many works where the strategy for active selection of action is guided by entropy reduction [5][9][8]. However, its computational cost may hinder performance, specially in the case of using particle filters [5][8], so a smaller number of actions have to be considered. This problem may be alleviated by using alternatives such as a mixture of Kalman filters [9].

Our approach can also be seen as a particle filter, as we use kernel density estimation on the hypotheses space. However, instead of resampling the posterior at each step, we represent it by a product of kernel density estimates. We observed that the resampling process increases the entropy on the posterior, which may affect the choice of the best hypothesis or the stopping criteria. Despite the fact that a product of densities is computationally demanding, existing state of the art methods provide very fast implementations [10], especially for situations where the distributions do not have much overlap.

III. Methodology

In this section, we give an explanation of the adaptation problem, and provide a probabilistic interpretation and show how it can be computed incrementally.

A. Experimental scenario

Our experimental scenario is the interaction of a humanoid robot with a tactile interface. This interface presents a virtual keyboard that continuously reproduces musical notes based on the position where the robot finger is located. The properties of the sounds that can be controlled are the tempo and the pitch of the notes.
The robot receives no visual information but a sound representation of the musical note being played and the proprioceptive information of where its finger is located.

Therefore, the task of the robot is to produce a sequence of musical notes by means of moving its finger across the virtual keyboard interface, as depicted in Figure 2. Hence, the actions are defined as reaching movements to a desired location in the robot working space.

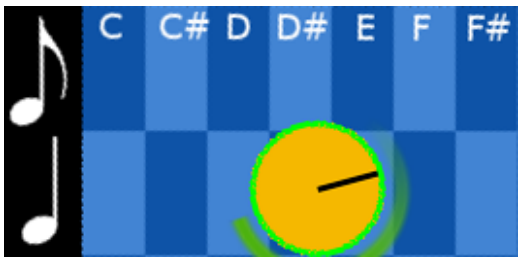

Fig. 2. Virtual keyboard interface for music interaction. The object, shown as a yellow circle, can be moved around by dragging it using the finger. Each cell changes both the note produced and the tempo in which it is emitted.

It can be noted that this setup can be extended to tasks which can be decomposed as sequences of atomic sub-tasks, e.g. writing a word, which is composed of chaining the writing of individual letters.
An example can be the sequence S_N of N sounds s_i represented as follows:

$$\mathbf{S}_N = \{s_0, ..., s_{N-1}\} \quad (1)$$

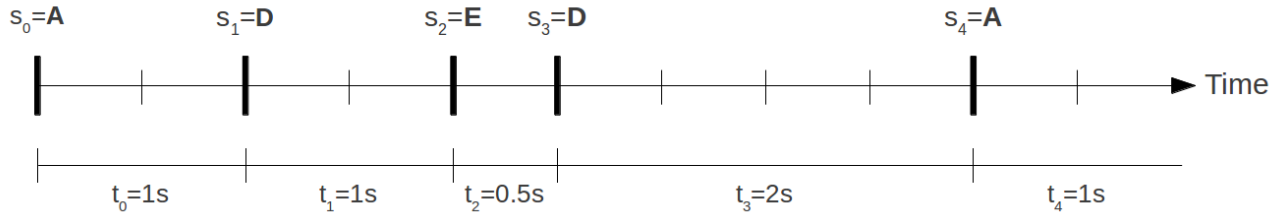

Fig. 3. Temporal representation of a sequence of musical events of the form $S = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_3), (s_4, t_4)\}$. The note is given by s_n, while the duration is given by t_n.

In Figure 3 we provide a temporal representation of an example note sequence of five pairs note-duration, $S = \{(A, 1), (D, 1), (E, 0.5), (D, 2), (A, 1)\}$.

In this way, the knowledge of the robot does not represent tasks but their component sub-tasks. Hence, given a task, its behaviour consists of segmenting a task in its individual sub-tasks and then executing the actions that enable the robot to perform each of the sub-tasks.

When interacting with an external tool, usually the robot must have a calibration between its internal and external working space. In our case, this means that the robot acquired knowledge assumes that the relative position between its body and the instrument does not change.

This is evident in the situation of a blind human pianist who is put every time in the same position in front of the piano. His knowledge is based purely on the positions of his fingers and the sounds perceived afterwards.
Following this line of thought, a question and a potential problem arises within this context. What happens if the pianist is not placed in the very same position? Then, it is obvious that they must undergo a process of *global adaptation* or recalibration of their hands with some mental references to where the notes are located in the keyboard based on their prior knowledge.

Turning back to our pianist robot, we cannot guarantee that each time it will be placed in the same position as when its model was acquired. This not only happens due to errors in the relative position of the instrument and the robot body, but also due to the complexities in the robot body or even failures in the robot.

In the case of a misplacement of the virtual keyboard relative to the robot body, this basically induces a transformation in the action space, as now the robot must adapt its actions to conform to the new positioning of the instrument.

Let us now introduce the notation used in our experimental setup. In our musical scenario, the task carried out by the robot is to produce a sequence of musical notes $S_N = \{s_i\}\, s.t.\, i \quad 0..N-1$. This is achieved by executing a sequence of actions a_i, each of which causing the instrument to play the required musical notes s_i for the specific durations.

We model the robot knowledge about the musical instrument probabilistically as the joint distribution of sounds and actions, i.e. $p(sound, action)$. When the robot wants to produce a particular sound s_i, the needed action a_i can be inferred using the conditional distribution, i.e. a_i $p(action|sound = s_i)$.

This work assumes that the learning of the model $p(sound, action)$ is already performed. Particularly, in our experiments we follow the active learning methodology described in [11], although other learning algorithms could be applied.

The action space is the space of positions in the workspace of the robot arm lying on top of the surface of the tactile interface. Given the continuous nature of this action space and the fact that each of the virtual keys correspond to a region in the interface and not a point, as can be seen in Figure 2, the mapping from sounds to actions is highly redundant, that is, there are many actions producing the same sound. This fact makes the recalibration of the action space a non-trivial problem, as we will show later.

In execution mode, once the robot perceives a sound, it has to generate an action and execute it in order to produce the next musical note in the sequence. The robot goes back to the first note once it reaches the last one, so the sequence is repeated in a loop. At time t, given the sound perception s_t, the sequence of musical notes S_N and the learned model of the instrument $\mathcal{M}_{INST}$, the robot generates the action depending on the result of the previous one. Algorithm 1 highlights the main steps involved in the generation of the next action to be executed.

Algorithm 1 Generate action for current perceived sound

1: **Input:** s_t, S_N, $\mathcal{M}_{INST}$, $history$
2: **Output:** $history$
3: i $\arg\min_i err(s_t, s_i)$ s_i S_N
4: **if** $err(s_t, s_i) <$ **then**
5: $nextSoundID$ $(i+1)\, mod\, N$
6: $addToHistory(MATCH)$
7: **else**
8: $nextSoundID$ 0
9: $addToHistory(NO\_MATCH)$
10: a_t $sampleAction(s_{nextSoundID},\ \mathcal{M}_{INST})$
11: Send action based on a_t

For further information, we refer the reader to [11] for more details into how the action is sampled from a given

sound s_i. The function $err(s_a, s_b)$ is an error function used to match two D-dimensional sound feature representations, and is an error threshold used to discard false matches. In our experiments, this function is the Manhattan distance for D-dimensional vectors.

$$err(a, b) = \sum_{i=1}^{D} |a_i - b_i| \quad (2)$$

B. Problem definition

In the model used for the experiments, $p(S, A)$, the action space A is defined as the Cartesian space $\mathbb{R}^2$ of positions where the finger of the robot is directed to interact with the musical instrument.

The position of the instrument relative to the robot is considered the same during the whole learning process. However, after the model is learned, the instrument can be displaced. This kind of situation may arise if the acquired model is reused for several experiments and the experimental scenario needs to be set up again, that is, we cannot assume that the instrument and the robot are placed in the same relative positions.

This means that the action space suffers a transformation $\mathcal{T}$, so the new distribution with the updated action space A can be expressed as:

$$p(S, A\ ) = p(S, \mathcal{T}\ (A)) \quad (3)$$

Hence, the problem here is to find the parameters $\hat{}$ of this transformation. In our experiments, we assumed that the transformation is global, that is, the optimal parameters $\hat{}$ are the same for the whole space A. Also, we restricted the family of transformations to translations of the action space, thus, $= \ _x, \ _y$, where $_x$ and $_y$ are displacements in the x and y coordinates, respectively. Note that this choice was motivated by the type of problem we deal with, and that more complex parametrization can be used to adapt to the kind of action space or transformations in the problem at hand.

C. Failure detection

As we showed in Algorithm 1, the robot maintains a history H of successful and failed actions in order to have empirical information about its current performance. At time t, each element H_i, equals to 1 if the action at time $t - i$ failed, and 0 in case the action was successful. By successful action, we mean that the action ended up producing the expected sound perception.

First of all, it has to be noted that the algorithm for failure detection herein proposed is not very complex, as it is not the purpose of this work to improve on existing detection methods. Given the nature of our data and the fact that the changes in performance are quite big for the kinds of failures that happen in this experimental scenario, we consider the following method to be adequate to our needs. However, other change detection algorithms could be used [12][13][14].

This history H is partitioned in two segments, a short-term memory composed of the most recent N^{ST} action results and a long-term memory which contains the older N^{LT} elements of the history. This partitioning is illustrated in Figure 4, and is used to perform failure detection. If we treat each movement as an independent Bernoulli experiment, from the history of the last N^{LT} action results, we can establish the probability of failing an action as:

$$p_{fail} = \frac{1}{N^{LT}} \sum_{i=1}^{N^{LT}} H_{i+N^{ST}} \quad (4)$$

Then we can have an event detection filter by looking at the short-term memory. We assume that the number of recently failed actions, defined as:

$$n_f = \sum_{i=1}^{N^{ST}} H_i \quad (5)$$

follows a Binomial distribution. Then, the probability of observing n_f failures in the recent history of action results is:

$$P(n_f; N^{ST}, p_{fail}) = \binom{N^{ST}}{n_f} p_{fail}^{n_f} (1 - p_{fail})^{N^{ST} - n_f} \quad (6)$$

If this probability falls below a certain threshold, a failure event is signalled and the adaptation process is started.

D. Adaptation of action space

Once a failure is detected, the robot enters into a re-calibration mode, which consists in adapting the action space represented by the model to the new one after the perturbation occurred.

Given that the perturbation is modelled as a transformation $\mathcal{T}\ (A)$ with parameters , the adaptation process is defined as an incremental estimation of these parameters by making use of kernel density estimation to represent the hypotheses space.

We are interested in the recovery of the solution as quickly as possible, that is, using only a few samples. For this reason, later we present an active version of this adaptation process, which is devised as an improvement over the baseline method.

Our adaptation process is similar to the methodology used in [15] for object localization in images, termed *Implicit Shape Model* (ISM). In that paper, authors find an object in a novel image by making use of kernel density estimation to find the parameters that best describe the transformation between the model of the object and the novel image. Every part of the object located in the new image casts votes for the object center and scale in a parameter space, so with sufficient votes, they are able to recover the most likely bounding box enclosing the object.

However, our approach differs in two important ways. First, our method is online, so our processing is done in a sequential way, which enables us to formulate an active

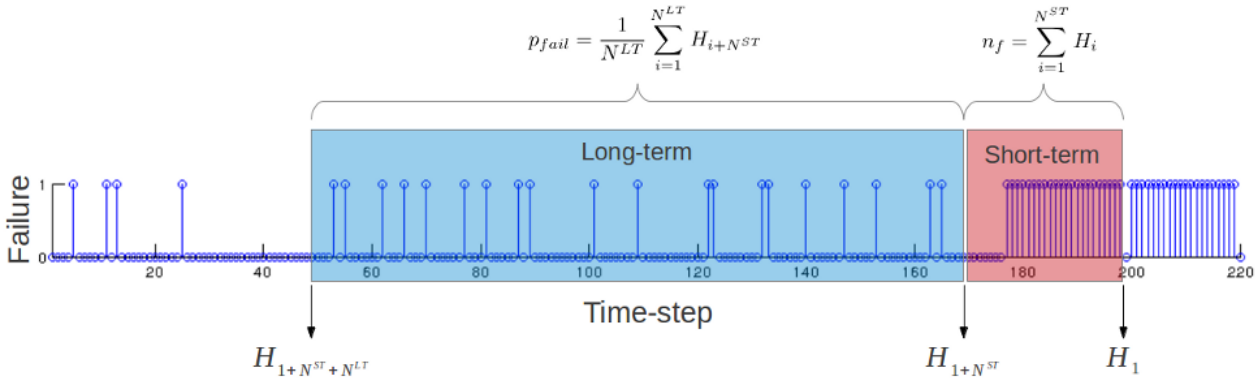

Fig. 4. Failure detection mechanism, assuming currently time is $t = 200$. History is divided in two segments. The short-term segment contains N^{ST} elements and is used to compute the number of failures n_f, while the long-term segment contains N^{LT} elements and is used to compute the probability of failure p_{fail}. These two quantities are used to compute the probability of a failure having occurred within the window $[t - N^{ST},\ t]$.

policy for choosing the next best sample to query, as will be shown later.

Secondly, we realised that the ISM method tends to produce a high number of false positives. This is due to the fact that the votes are aggregated by a summation of their support regions, rather than a multiplication.

This later fact is what motivates our approach. In our case, at each time step, the robot receives a pair (s_t, a_t), where s_t is the sound perceived after the execution of action a_t. After the transformation in the action space $\mathcal{T}(A)$ occurred, the sound s_t might not be the same as the expected sound when the action a_t was executed. Assuming that the adaptation process is activated, each new perception (s_t, a_t), when related to the learned model $\mathcal{M}_{INST}$, contains information about the distribution of parameters for the transformation $p(\ |s_t, a_t, \mathcal{M}_{INST})$.

Algorithm 2 Generate a hypotheses space for a new sample

1: **Input:** s_t, a_t, $\mathcal{M}_{INST}$
2: **Output:** KDE_t
3: $\mathbf{a}$ $sampleFrom(p(a|s_t, \mathcal{M}_{INST}))$
4: $\mathbf{h}$ []
5: **for** a_i **in** $\mathbf{a}$ **do**
6: h_i $computeTransform(a_i, a_t)$
7: KDE_t $buildKDEFromSamples(\mathbf{h})$

We approximate this distribution by sampling from the model and represent the hypotheses space with a kernel density estimate. Algorithm 2 highlights the main steps involved in its computation. We obtain a sample of the actions $\mathbf{a}$ required to obtain the perceived sound s_t. This action set $\mathbf{a}$ generates a set of hypotheses $\mathbf{h}$ based on a_t. Each of the hypotheses h_i in the set are the parameters $_x$, $_y$ that transform the action a_i into the previously executed action a_t.

A KDE is then built from this hypotheses set, which represents the distribution of hypotheses for the sample (s_t, a_t) in the following form:

$$p(\ |s_t, a_t) = \sum_i^N K \ (\ - h_i) \tag{7}$$

where K is the kernel function used for the estimation and its parameters. For the sake of simplicity, we have obviated the dependency on the model from this equation, but the reader must consider that all the sampling is obtained from the learned model.

Now, given a sequence of independent actions $\mathbf{a} = (a_1, a_2, ..., a_t)$ up to time t, the distribution of the hypotheses space for the transformation parameters given the sequence of actions is:

$$p(\ |s_1, a_1, ..., s_t, a_t) = \prod_{i=1}^{t} p(\ |s_t, a_t) \tag{8}$$

Note here that the main difference between our approach and the work by Leibe et al. [15] is in the aggregation of the individual estimates for $p(\ |s_t, a_t)$, which in our case are multiplied together. In cases where the number of inliers per hypothesis is very low, multiplication of the individual hypotheses distributions yields better results than summation. Furthermore, the effects of outliers are magnified by summation.

The key computation in this approach is the product of kernel density estimates. Albeit this is not a trivial task, recent methods enable the sampling from a product of kernel density estimates in an efficient way [10].

At each time-step, we can compute the parameters which have the maximum likelihood in the current hypotheses space and stop the process when it reaches a certain threshold. Other stopping criteria may also be used, like the rate of reduction in the entropy of the distribution or a comparison with the second best hypothesis, i.e. the amount of confusion between competing hypotheses. However, the criterion used here was found to work well in practice.

E. Active adaptation strategy

The active strategy presented in this paper is based on the selection of actions which are expected to reduce the entropy of the hypotheses distribution. In our case, we use a simulation process which first samples a series of candidate actions and then chooses the best one by computing the expected reduction in entropy if such an action is executed.

In order to further reduce the computational effort needed in this step, we introduced a heuristic to select candidate actions. It is based on the fact that looking for the boundaries between two regions of the action space producing different sound yields better results. This intuition was found after questioning a few people how would they solve the task and then also corroborated empirically.

After a number of candidate actions are sampled from the model, we compute their confusion factor, which is the uncertainty of the model over which sound is heard after executing that candidate action. Then, we retain the most confusing ones for further evaluation.

The selected candidate actions are now evaluated in terms of the expected entropy reduction on the distribution over hypotheses. We do this by first simulating the execution of the action given the best hypothesis so far. Then, the expected action results are used to update the current hypotheses distribution and the entropy reduction can be computed. This shares some similarities with the candidate selection process applied in [11], which yielded good results in terms of learning speed.

IV. Experimental Results

In order to test the adaptation method without the uncertainties introduced by the dynamics of the robot simulator or the real iCub itself, we first tested our algorithm in an idealized problem. That is, we removed the need for a dynamical system – real or simulated – to execute the action and reach for a particular location in the virtual keyboard, so the action always reached the intended location with no uncertainty. It also enabled for faster experimentation and clarity in the interpretation of results.

A. Comparison of two strategies

We wanted to know how the hypotheses space distributions evolve as we incorporate more data in the incremental estimation. The comparison is between a baseline strategy, consisting in the random selection of samples and the active selection, were we simulate the expected effects of the sample in the hypotheses distribution, and selecting the one that is expected to provide the most information.

In Figure 5 we show the results for an example experiment using the baseline and the active strategies. The top row shows the aggregated information between all the samples up to the indicated time step. It can be appreciated that due to the sampling process performed in the baseline strategy, the information introduced when aggregating together the hypotheses generated at each time step is not very high.

However, in the bottom row we show the results for another example experiment using the active strategy. In this case, the sampling considers the aggregated distribution until the current time step to decide which sample to query, so when aggregated together, the entropy of the resulting distributions is reduced very quickly.

Figure 6 shows this decrease in entropy for a series of experiments comparing both approaches. It can be seen how the active selection of samples exhibits a quicker decrease in the entropy, hence the most likely hypothesis can be selected earlier and more confidently.

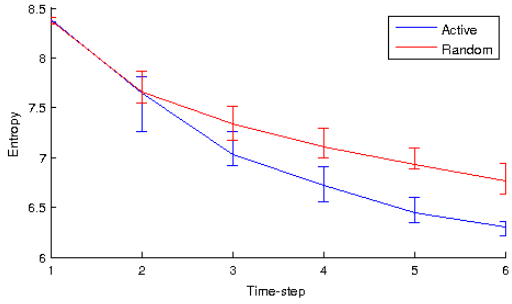

Fig. 6. Comparison of the entropy reduction between both adaptation strategies. Note how the active selection of samples exhibits a quicker decrease in the entropy, hence the most likely hypothesis can be selected earlier and with more confidence.

B. Estimation error

We evaluate how the estimation error of the transformation parameters decreases as we introduce more observations. We tried different parameters, in order to see the benefits of the active strategy depending on the problem. For each transform parametrization, we performed 50 experiments. Then, we clustered together the different parametrizations in three categories, in order to compare the benefits of the active strategy when the transformation is a small, medium or large displacement. These categories refer to displacements up to 10, 20 and 40 pixels, respectively. It has to be noted that each key of the virtual keyboard spans a region of 50 by 50 pixels.

For each experiment, we also extracted how many samples are needed to obtain a root-mean-square error (RMSE) below a particular threshold. Finally, we ran a Wilcoxon rank sum test and obtained the significance results for the different cases.

Figure 7 shows the results for the three types of transforms. For small displacements, the active strategy clearly outperforms the baseline, both in terms of convergence speed and final RMSE. The p-value was below 10^{-5} for RMSE up to 10 pixels. For medium displacements, the difference is not so big, although still significant – p-value below 10^{-3} for RMSE up to 8 pixels –, specially in the variance of error at convergece. However, for large displacements, the improvement is not very significant.

C. Results with robot dynamics

After successful evaluation of the adaptation method, we incorporated it to the robotic platform in order to see how the

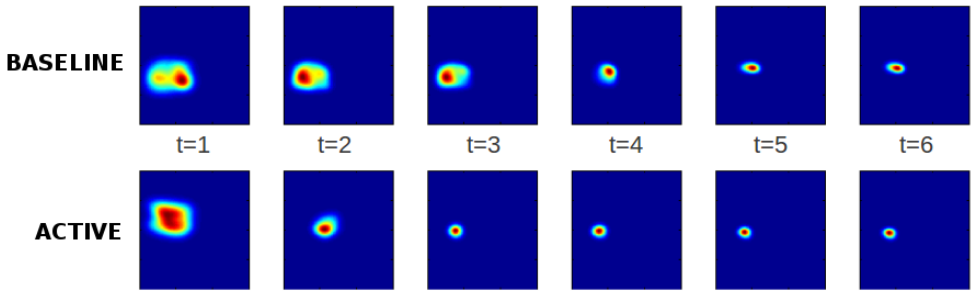

Fig. 5. Evolution of the parameter space for the random strategy.

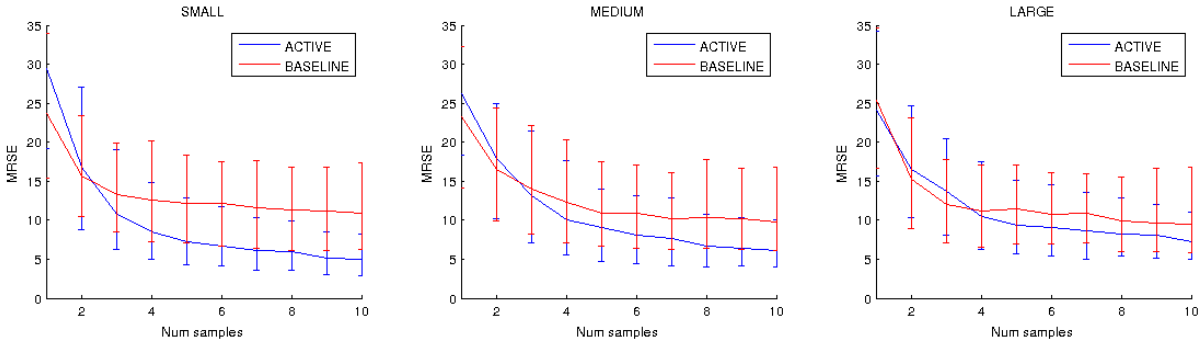

Fig. 7. Error results for three problem categories. For small displacements, the active strategy clearly outperforms the baseline, both in terms of convergence speed and final RMSE. For medium displacements, the difference is not so big, although still significant, specially in the variance of error at convergence. However, for large displacements, the improvement is not very significant.

dynamics of the simulated and real robot affect the system performance.

The robot correctly adapted to the perturbations introduced by the transforms, restoring its previous performance after the execution of the proposed algorithm. However, although our results point in the direction of the same kind of improvement as in the results shown in the previous section, the differences between both strategies were not significant enough.

We believe that one of the main reasons was that the model learned with the iCub robot was not accurate enough and the possible advantage of using the model-based active strategy was hindered by the low quality of the model itself. More experiments are needed to confirm this hypothesis.

V. Conclusions

In this paper we introduced a method for adaptation of the action space of a robot when a perturbation is introduced in its environment. We exemplified this situation with a humanoid robot interacting with a musical instrument, where the perturbation is a displacement of the keyboard the robot is playing with.

Given that the robot has a model of the locations of the different sounds in the keyboard, it can exploit this knowledge to perform an adaptation in an incremental way. After a few trials, the robot successfully finds the displacement of the virtual keyboard so its performance is restored to its previous level.

As in the field of robot global localization, the task here is to estimate the posterior over the locations in a map – or in our case, the displacement of the keyboard – given a sequence of observations.

The main novelty introduced here is the representation of this posterior as a product of kernel density estimates, computed using a state of the art method. This alleviated the negative effect on the posterior distribution entropy introduced by the resampling process of a typical particle filter.

We presented the results first in an idealized problem, where we removed the robot action execution dynamics, so we could focus in studying the benefits of active action selection. We obtained promising results, particularly when the magnitude of the displacement is small, compared to big displacements.

However, when testing the algorithm in the robot, the results were not significantly better when comparing the active strategy against the baseline. We found that the dynamics of the action execution and probably a learned model which did accurately represent the underlying sensorimotor map played an important part in the obtained results.

Notwithstanding, the method itself works well in practice and it is the active strategy which needs more experimentation in order to confirm the results obtained in the idealized problem. Future work aims at using a better sensorimotor model, as well as more extensive evaluation.

ACKNOWLEDGMENT

This work was supported in part by the Generalitat de Catalunya to Consolidated Groups 2014 SGR Grant 118, the CSIC intramural project 201250E054 and the EU FP7 Project WYSIWYD under Grant 612139. It has also received support from the COR (TIN2012-38876-C02-01) project.

REFERENCES

[1] A. Stoytchev, "Some basic principles of developmental robotics," *Autonomous Mental Development, IEEE Transactions on*, vol. 1, no. 2, pp. 122–130, 2009.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[3] A. C. Murtra, J. M. M. Tur, and A. Sanfeliu, "Efficient active global localization for mobile robots operating in large and cooperative environments," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 2758–2763.

[4] A. J. Davison and N. Kita, "3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–384.

[5] J. M. Porta, J. J. Verbeek, and B. J. Kröse, "Active appearance-based robot localization using stereo vision," *Autonomous Robots*, vol. 18, no. 1, pp. 59–80, 2005.

[6] P. Jensfelt and S. Kristensen, "Active global localization for a mobile robot using multiple hypothesis tracking," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 5, pp. 748–760, 2001.

[7] A. Andreopoulos and J. K. Tsotsos, "Active vision for door localization and door opening using playbot: A computer controlled wheelchair for people with mobility impairments," in *Computer and Robot Vision, 2008. CRV'08. Canadian Conference on*. IEEE, 2008, pp. 3–10.

[8] M. Renfrew, Z. Bai, and M. C. Cavusoglu, "Particle filter based active localization of target and needle in robotic image-guided intervention systems," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*. IEEE, 2013, pp. 448–454.

[9] D. Ognibene and Y. Demiris, "Towards active event recognition," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 2495–2501.

[10] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," *Communications of the ACM*, vol. 53, no. 10, pp. 95–103, 2010.

[11] A. Ribes, J. Cerquides, Y. Demiris, and R. Lopez de Mantaras, "Active learning of object and body models with time constraints on a humanoid robot," *Autonomous Mental Development, IEEE Transactions on*, vol. 7, no. 3, 2015.

[12] M. Markou and S. Singh, "Novelty detection: a reviewpart 1: statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[13] G. D. Breda and L. d. S. Mendes, "Qos monitoring and failure detection," in *Telecommunications Symposium, 2006 International*. IEEE, 2006, pp. 243–248.

[14] S. A. McKenna, D. Hart, K. Klise, V. Cruz, and M. Wilson, "Event detection from water quality time series," in *Proc. ASCE World Envir. & Water Resources Congress, Tampa, Fla*, 2007.

[15] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 259–289, 2008.