2012 IEEE RO-MAN: The 21st IEEE International Symposium on
Robot and Human Interactive Communication.
September 9-13, 2012. Paris, France.

# Online Customization of Teleoperation Interfaces

Anca D. Dragan and Siddhartha S. Srinivasa
The Robotics Institute, Carnegie Mellon University
{adragan,siddh}@cs.cmu.edu

***Abstract*—In teleoperation, the user's input is mapped onto the robot via a motion retargetting function. This function must differ between robots because of their different kinematics, between users because of their different preferences, and even between tasks that the users perform with the robot. Our work enables users to *customize* this retargetting function, and achieve any of these required differences. In our approach, the robot starts with an initial function. As the user teleoperates the robot, he can pause and provide example correspondences, which instantly update the retargetting function. We select the algorithm underlying these updates by formulating the problem as an instance of online function approximation. The problem's requirements, as well as the semantics and constraints of motion retargetting, lead to an extension of Online Learning with Kernel Machines in which the width of the kernel can vary. Our central hypothesis is that this method enables users to train retargetting functions to good outcomes. We validate this hypothesis in a user study, which also reveals the importance of providing users with tools to verify their examples: much like an actor needs a mirror to verify his pose, a user needs to verify his input before providing an example. We conclude with a demonstration from an expert user that shows the potential of the method for achieving more sophisticated customization that makes particular tasks easier to complete, once users get expertise with the system.**

## I. Introduction

When teleoperating a robot (or an animated character), we rely on a *retargetting function* – a function that maps the user's input onto the robot. This function must be different for different robots, because of their variation in kinematics [1], [2]. Similarly, it must be different for different users, because of their variation in preferences: not every user will want the same retargetting function, much like how not every user likes the same graphical interface [3]–[5]. Finally, the retargetting function must be different for different tasks, e.g. pushing the buttons on the microwave panel requires more finesse than waving an arm in greeting, and even at different configurations for the same task, e.g. grasping an object to the far left, near joint limits, is more challenging than grasping it in the middle of the workspace. In this work, we explore the idea of enabling users to customize the retargetting function. Our method can be used to adapt any prior function to the three factors stated above: the kinematics of a particular robot, the user's preferences, or the specific task at hand.

The customization process, outlined in Fig.2, happens online. The robot starts with an initial retargetting function, i.e. it has a *prior*. As the user teleoperates, he pauses the robot and provides example correspondences mapping user input to robot pose. Each example instantly updates the current

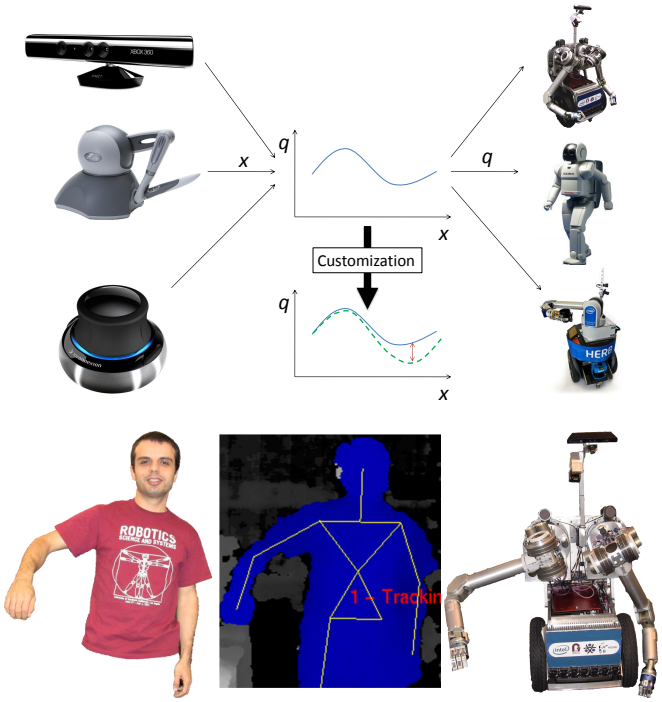


Fig. 1. We propose a method for customizing the retargetting function which maps the user's input onto the robot during teleoperaion.

retargetting function, enabling the user to base his future examples on the outcome of previous ones. In doing so, our work bridges two paradigms, one stemming from graphics and animation, and the other from teleoperation interfaces.

In the animation community, motion retargetting is used to map an actor's trajectory onto an animated character (e.g. [1], [2], [6], [7]). Although the traditional way to do so is via constrained optimization [1], recent work has proposed to learn a retargetting function offline, based on key pose correspondences [6], or based on a different, manually retargetted trajectory [7]. Our approach extends this idea to an online setting, where users can provide examples by teleoperating the robot or character using the current version of the retargetting function.

In the teleoperation community, unlike in animation, motion retargetting happens in real-time, and across the entire space rather than along a pre-set trajectory. While there are many interfaces for teleoperation, especially for robot manipulators [8]–[10], the retargetting function is usually designed a-priori and cannot be changed. Some robots, however, such as the DaVinci from Intuitive Surgical [11], allow users to change the retargetting function online, albeit

only by a constant (or global) offset, equal throughout the input space. The surgeon operating the DaVinci can use a clutch to stop teleoperation and reset himself to a comfortable position [11], thus offsetting the origin of the retargetting function. Our approach extends this idea to variable offsets: the user can modify different parts of the space in different ways, and can reach *any* (smooth) retargetting function.

Our method is an instance of Learning from Demonstration [12], where the robot learns a policy based on examples. A significant amount of work in this area [13]–[15] has applied function approximation techniques to learn the policy from examples provided by a human or another robot.[1] We adopt the same principle, and use an extension to Online Learning with Kernel Machines [16] as the function approximator.

We test our central hypothesis – *that our method enables novice users to customize prior retargetting functions to good outcomes* – in a user study. The results confirm the hypothesis, and point the way to future improvements. One of our users had a particularly hard time providing good examples: although he did improve a bad prior in one condition, the examples he gave to a good prior in a different condition made this prior drastically worse. Our analysis revealed that a major problem was the lack of visual feedback on his input: much like an actor needs a mirror to verify his pose, a user needs to verify his input before providing an example. Overall, we believe that the ability to customize retargetting functions online is key to a better teleoperation experience. We are excited about pursuing this research further, by developing better ways of giving feedback to the users, making the effects of examples more intuitive, and exploring methods for generalizing customized retargetting functions to new tasks.

## II. Customizing the Retargetting Function

Our goal is to enable users to produce a retargetting function $f : x \mapsto q$ that maps their pose $x$ to a configuration for the robot, $q$. We assume we are given a smooth prior $f_{prior}(x)$, and decompose $f(x)$ into this prior and an offset function: $f(x) = f_{prior}(x) + o(x)$. As the user provides examples, our method trains the retargetting function to reflect his preferences by learning $o(x)$ online, via function approximation.

In this section, we outline the requirements of this training process and recap an online learning algorithm from the machine learning literature that meets these requirements. Finally, we present a novel modification to the algorithm that incorporates domain knowledge to approximate retargetting functions with fewer examples.

### A. Problem Requirements

1) **Online:** The user should be able to provide further examples based on the outcome of the previous examples. Therefore, changes should be incorporated online, and in real-time.
2) **Smooth:** The function $o(x)$ needs to be smooth, as there can be no sudden shifts that cause the robot to move abruptly while the user is barely changing pose.
3) **Propagation decay:** Different areas of the input space require different offsets. Therefore, examples typically have local support and should decay as we move away from them in the input space to allow different alterations in different areas.

[1]The motion retargetting problem is also related to the correspondence problem [12]: that of mapping a demonstration onto a robot. The latter, however, goes beyond motion, to task-level differences, sensing differences, and capability differences in general.

### B. A Recap of Online Learning with Kernel Machines

The requirements above suggest using an algorithm that alters the function with every example in a smooth way, propagating the changes to the rest of the space. For these reasons, we chose to use Kernel Machines and perform online least squares in a functional space, an algorithm first introduced by Kivinen [16]. The algorithm satisfies the requirements from above: if works well online, smoothness in ensured by the function representation, and propagation decay is the result of the gradient of the loss function. We describe the algorithm here for completeness. Further details can be found in [16].

*1) Function Representation:* The premise of the algorithm is to represent the function to be approximated (in our case $o(x)$), as a point in a Reproducing Kernel Hilbert Space (RKHS) of functions $(\mathcal{H}, ||\cdot||_{\mathcal{H}})$:

$$o(x) = \sum_i \alpha_i k(x_i, x) \tag{1}$$

for some input poses $x_i$, and $\alpha_i \in \mathbb{R}$. $k$ is a reproducing kernel function, and a common choice for $k$ is a Radial Basis Function (RBF):

$$k(x_i, x) = \exp\left(-\frac{1}{2}w(x_i - x)^2\right) \tag{2}$$

where $w \in \mathbb{R}$ determines the width of the kernel. The smoothness of the kernel guarantees the smoothness of the retargetting function.

*2) Online Learning:* Given our representation, we want to find the function that best fits the user examples: a minimization of a loss function over the space of representations. We choose a typical loss function composed of the sum squared error between the user examples and the prediction, subject to regularization:

$$\min_o L = \min_o \sum_i L_i = \min_o \sum_i \frac{1}{2}(q_i - o(x_i))^2 + \frac{1}{2}\lambda||o||^2_{\mathcal{H}} \tag{3}$$

where the pair $(x_i, q_i)$ represents example $i$.

To solve this online,[2] Kivinen [16] proposed using stochastic gradient descent – minimization by descending the gradient of the instantaneous loss $L_i$, rather than the cumulative

[2]For a small number of user examples, it is feasible to run the batch version for every new example.

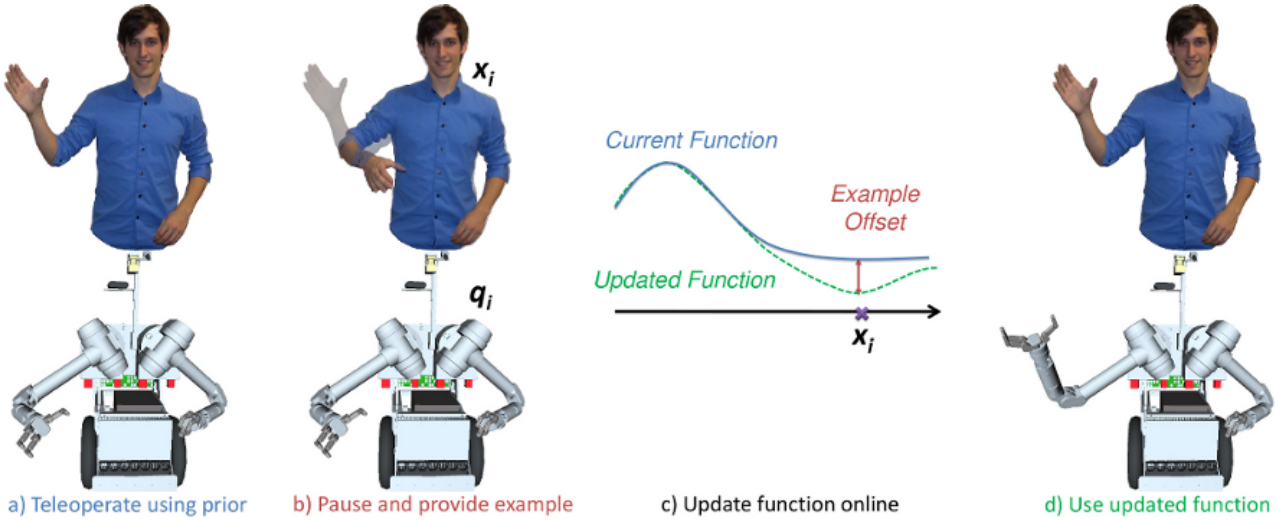


Fig. 2. The training process (showing the user giving one example to the learner).

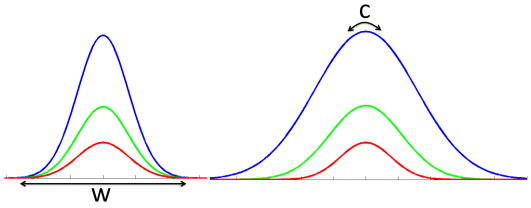


Fig. 3. Comparison of kernels with fixed width vs. fixed max curvature. Fixing the maximum curvature makes larger changes propagate further.

loss $L$. Therefore, as each example is received, the offset is updated as:

$$o_{i+1} = o_i - \eta \nabla L_i \tag{4}$$

where

$$\nabla L_i = -(q_i - o(x_i))k(x_i, \cdot) + \lambda o \tag{5}$$

The algorithm proceeds as follows:

RBF-LEAST-SQUARES

1 $o \leftarrow 0$
2 **for** $i \leftarrow 1$ **to** $N$
3 **do** $\alpha_i \leftarrow q_i - o(x_i)$
4 $o \leftarrow (1 - \eta\lambda)o + \eta\alpha_i k(x_i, \cdot)$

With every new example $(x_i, q_i)$, the method discounts previous data and adds an additional term to the function, which places an offset at $x_i$ proportional to the desired change and propagates to the rest of the space according to the kernel width corresponding to $w$. Therefore, the algorithm satisfies all of the requirements from Section II-A, and scales linearly with the number of examples.

### C. Our Extension to Varying Kernel Widths

RBF-LEAST-SQUARES uses a fixed kernel width $w$ for all examples. The changes propagate in the same way to the rest of the space, independently of the nature of the change, as in Fig.3(left). However, two aspects of our problem make this undesirable: the semantics of changes to the prior, and constraints on the maximum acceleration of the robot.

First, large changes have different semantics than small changes when training the retargetting function. Usually, large changes need to be propagated wider through the space (i.e. with a very wide kernel), because they are meant to change the prior in a fundamental way – the prior is in some way offset by a considerable amount from the desired mapping. Small changes, however, should affect the space only locally (i.e. with a narrow kernel), because they usually represent local corrections.

Second, safety and hardware constraints suggest a bound on the acceleration of the robot. This is equivalent to bounding the curvature of the retargetting function.

Therefore, we have two additional requirements: larger changes must propagate further, and the maximum curvature has to be bound. We resolve these two requirements by keeping the curvature $c$ at the center of the kernels constant. Fig.3 shows that this enables us to enforce the first requirement: larger changes do propagate further. With respect to the curvature requirement, we assume that examples are sparse enough that the curvature at the center of the kernel is negligibly affected by other examples. Then, the maximum curvature of the offset function $o(x)$ occurs at an example point $x_i$, and is therefore $c$. As a result, the maximum curvature of $f(x)$ will only depend on $c$, and can be selected to satisfy the desired bound.

To determine the relation between the width and the maximum curvature, we define the constant maximum curvature constraint as

$$\frac{d^2}{dx^2} \alpha \exp\left(-\frac{1}{2}w(x_i - x)^2\right)\Big|_{x_i} = c, \quad \forall \alpha \in \mathbb{R} \tag{6}$$

Equivalently, the following equation must hold:

$$\alpha \exp\left(-\frac{1}{2}w(x_i - x)^2\right) w^2(x_i - x)^2 + \exp\left(-\frac{1}{2}w(x_i - x)^2\right) w\Big|_{x_i} = c \tag{7}$$

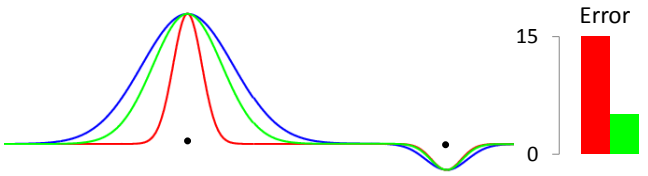


Fig. 4. An illustration of the fixed width kernels (red) vs. fixed max. curvature kernels (green) for approximating the function in blue after 2 examples.

Evaluating the Hessian at $x_i$ (the center of the kernel), we get that: $\alpha w = c$, or equivalently

$$w = \frac{c}{\alpha}$$

Fig.4 illustrates the propagations obtained from two examples (given at the black dots, from the function shown in blue) from the two versions of the algorithm: using fixed kernel widths vs. fixed maximum curvature. The maximum curvature version propagates the larger changes further, and conforms to the maximum acceleration bound.

## III. Empirical Results

Our empirical evaluation is focused on the question "Does our method enable novice users to successfully train prior retargetting functions?". We answer this in a study that tests the methods's usability, where we ask users familiar with the robot, but not with the teleoperation interface, to train a bad prior. We verify that the trained prior's performance reaches the level of a good retargetting function, showing that the training was successful. Aside from improving bad priors, our method also enables adaptation of good priors to particular tasks. Testing this particular aspect with novice users remains a future work goal. However, we show an example from an expert user on a difficult task, where the time taken to train the prior and complete the task is much lower that attempting the task without training.

### A. Usability: Users Can Fix Bad Priors

We designed a study to decide whether users would be able to repair a poor retargetting prior using our system. We used the HERB platform [17] for our experiments. HERB comprises of two Barrett WAM arms mounted on top of a Segway RMP 200 platform. The WAM arm joints are roughly anthropomorphic, with a total of 7DOFs made up of a 3DOF shoulder, a 1DOF elbow, and a 3DOF wrist. HERB's joint limits, as well as its link lengths and proportion, differ from those of an average human.

*1) Manipulated Variables:* We manipulated two factors: **training** (whether the user trains the prior or uses it as-is) and **prior quality** (which function acts as the prior). We used two priors in this study. The first one is a natural prior (which we call $NP$), obtained by taking advantage of HERB's fairly anthropomorphic kinematics: the function directly maps the operator's shoulder and elbow joint angles (obtained by processing the OpenNI (www.openni.org) skeletal tracker output) onto the robot. To manipulate the quality and create a bad prior, we added a large constant offset to the shoulder pitch: $UP(x) = NP(x) + c$. Fig.5 shows a user's configuration with its retargetted configuration on the robot via $UP$. This leads to four experimental conditions and four retargetting functions whose performances we compare: $NP$, $NP-T$, $UP$, and $UP-T$ (where $x-T$ refers to the function obtained by training the prior $x$).

*2) Subject Allocation:* We chose a within-subjects design, where each of our 6 users (all students at Carnegie Mellon, all familiar with HERB but not with the teleoperation interface) was assigned to all four conditions, in order to enable paired comparisons among the conditions. We first had an introductory phase, where the users interacted with the natural prior $NP$. We then tasked the users with training both priors. We instructed them to train the functions for general purposes, thinking about any task they might want to achieve with the robot, and stop when they are satisfied with the retargetting function they reached. After training, we moved the robot in front of a table and tasked the users with testing each of the four functions (the two untrained and the two trained) on a simple manipulation task – grasping a bottle on the table. To avoid ordering effects, we randomized the order of the conditions. Each user attempted each task three times, in order to reduce the variance in the data.

*3) Dependent Variables:* For each condition, we measured the **success rate** and the **time** to completion for the grasping attempts. Potential failures included knocking over the target object, colliding with the table, or being unable to complete the task within 200 seconds.

### B. Hypotheses

We first want to show that we manipulated the priors correctly: H1. *Prior quality has a significant positive effect on the dependent variables.*

Our second hypothesis is that training improves overall performance: H2. *Training has a significant positive effect on the dependent variables.*

Finally, we want to demonstrate that the trained unnatural prior does not perform worse than the natural prior and the trained natural prior. We define "does not perform worse than" using the concept of "non-inferiority" [18], [19]: a distribution is non-inferior to another if their difference is significantly greater than a negative margin. H3. *$UP-T$ is non-inferior to $NP$ and $NP-T$.*

### C. Analysis

We performed a factorial repeated-measures ANOVA on the success rate with Tukey corrections for multiple comparisons and a significance threshold of p = 0.05, which resulted in a good fit of the data ($R^2 = 0.845$). In line with our first two hypotheses, we found that both factors had significant effects. Prior quality improved the success rate by 30.6% ($F(1,20) = 22.07$, $p < .001$), and training resulted in an average improvement of 40.1% ($F(1,20) = 37.9$, $p < 0.001$). The interaction term was also significant ($F(1,20) = 49.02$, $p < .001$), the post-hoc analysis revealing that indeed, $UP$ was significantly worse than the other three conditions. The

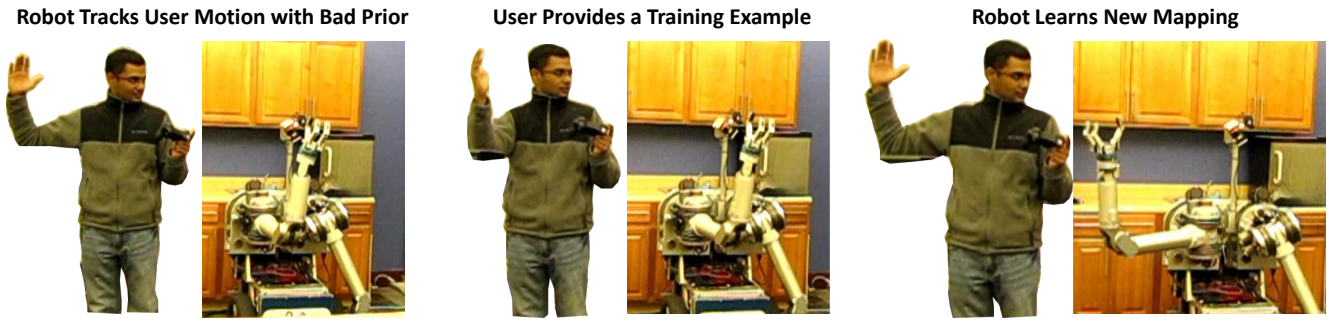


Fig. 5. Training with an unnatural prior.

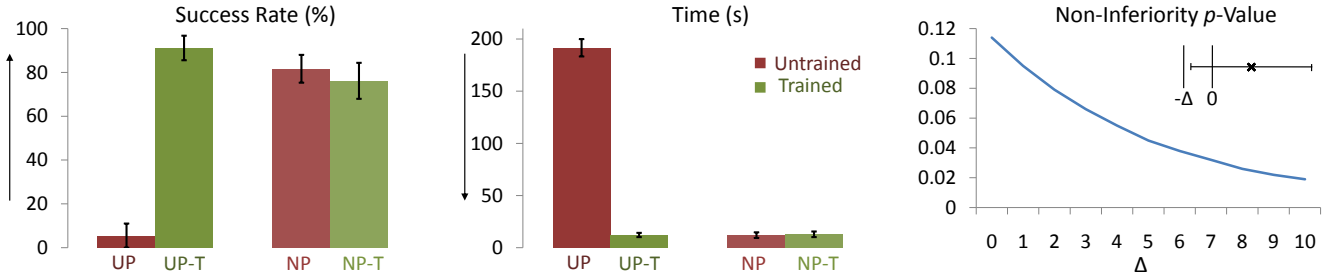


Fig. 6. Left: The mean success rate on within condition. Center: The mean time to completion within each condition. Right: $p$-values for non-inferiority tests with different margins. As the margin increases, we are more and more confident that the difference between $UP-T$ and $NP$ is greater than the negative margin. The same figure also depicts a typical non-inferiority test, where the lower bound for the 95% confidence interval must be greater than the negative margin.

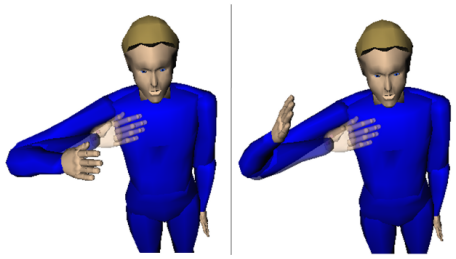

Fig. 7. The user's poor self-perception leads to poor examples. This is a focus side-effect: focusing on fixing a particular problem sometimes introduces undesired additional offsets. On the left, the approximate intended example, and on the right the provided example, which adds a large offset to the roll.

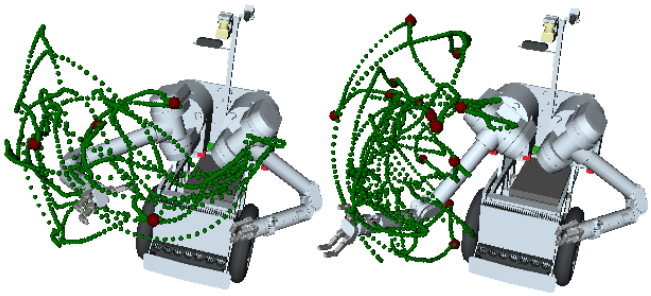

Fig. 8. The space two users explored while training. The green curves trace the end effector of the robot. The red points are locations of examples. The robot is positioned in the last example's configuration.

timing results were analogous, and Fig.6 plots the means for both success and time over the four conditions. We attribute the slight mean decrease in success rate of $NP-T$ to noise.

The largest outlier in our ANOVA was a user who performed much worse on the trained natural prior, $NP-T$, than on the untrained prior $NP$. This user commented during the study that proprioception was an issue ("It's really difficult sometimes because I am not good at perceiving myself"). Indeed, *poor perception of the input causes users to provide poor examples*, especially when focusing on fixing a particular problem in the prior and introducing undesired offsets in other dimensions. An example is in Fig.7: the user was fixing a problem with the elbow angle, but introduces a large offset in the shoulder roll with his example.

Proprioception was not the only factor contributing to this user's poor training performance. Fig.8 shows the difference between this user and second user, with better training performance: the latter explores more parts of the space, including configurations close to those necessary for the grasping task that served as a test after training, which puts him at an advantage. We see this as a side effect of not having told users what the task was ahead of time, in order to prevent them from training the priors only for the test task: a user that missed the task-related areas of the space performed worse on the task, *stressing the importance of task-specific retargetting functions*.

In terms of the number of examples, the users did provide significantly more examples to $UP$, as expected ($t(6) = 16.44$, $p < 0.001$): $UP$ received an average of 11 examples, with 4 being the minimum and 20 the maximum. Interestingly, not all users provided examples to the natural prior $NP$, some deeming it suitable as-is. However, there was no significant (or marginal) correlation between number of examples and performance: more examples did not mean

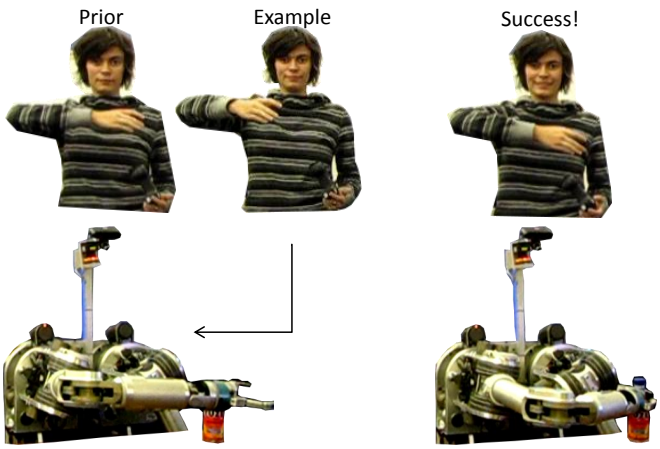


Fig. 9. Adapting a good prior to a specific task.

better/worse performance. The number of examples given to $NP$ significantly correlated to the one given to $UP$ (Pearson's $r(3) = 0.94$, $p = 0.004$), suggesting that some users are generally more lenient than others.

To confirm the last hypothesis, H3, we need to show that the difference in performance between $UP - T$ and $NP$ is significantly greater than a negative margin $-\Delta$. Setting $\Delta = 10\%$, a one-tailed paired $t$-test verified the hypothesis: $9.5 > -10$, $t(6) = 2.81$, $p = 0.019$. Therefore, *the trained bad prior is not worse, on average, by more than 10%*. Fig.6(right) provides a graphical explanation of the test, together with the $p$-values obtained from various margins between -10 and 0 (the larger the margin, the more confident we are that $UP-T$ is not worse by more than that margin). The result for $UP - T$ vs. $NP - T$ was similar: $t(6) = 4.34$, $p = 0.004$ for the same $\Delta = 10\%$

### D. The Method Enables Adaptation of Good Priors

Although $NP$ is a natural retargetting function in general, the kinematic differences between a human and HERB make particular tasks (such as grasping an object on the far left of the robot) particularly difficult. Our method enables the online adaptation of $NP$ to such tasks: Fig.9 shows that an expert user can provide an example that makes the task area easier to reach. Due to propagation decay, the example does not have undesired effects on other areas of the space (unlike a global offset technique, see Fig.10): *Trained retargetting functions must be more than global offset versions of their priors*. While an expert user was not able to succeed within 200 seconds with $NP$, it only took 27 seconds to provide the example *and* complete the task. This is in no way reflective of the performance of a novice user with the system – it is merely a vote of confidence in what the training method enables an experienced user to accomplish.

## IV. Conclusion

We introduced a method for enabling users to customize the motion retargetting function that maps their input onto the robot, by providing example correspondences during teleoperation. We validated its usability in a user study,

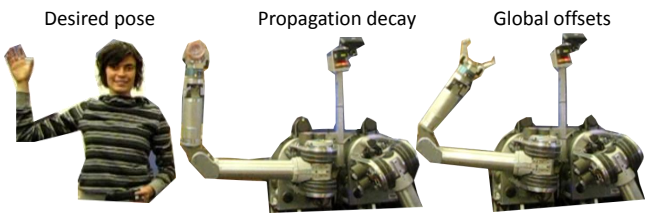


Fig. 10. Our method adapted the prior in the desired region of space, while leaving the rest of the space unaltered. If the propagation did not decay, i.e. the offsets were global, examples could have an undesired effect on the rest of the space.

which also pointed to a possible danger of the setup: poor proprioception can result in poor examples. In future work, we plan to explore the design of an interface that provides more feedback to the users as they give examples, and analyze long-term use effects. We are excited about a future where robots learn to better interface with their users, and believe that this work gets one step closer to that ideal.

## References

[1] M. Gleicher, "Retargetting motion to new characters," in *SIGGRAPH*, 1998.
[2] C. Hecker, B. Raabe, R. Enslow, J. DeWeese, J. Maynard, and K. Van Prooijen, "Real-time motion retargeting to highly varied user-created morphologies," in *SIGGRAPH*, 2008.
[3] L. Bergman, T. Lau, V. Castelli, and D. Oblinger, "Programming-by-demonstration for behavior-based user interface customization," *IBM Research Report*, 2004.
[4] S. Greenberg and M. Boyle, "Customizable physical interfaces for interacting with conventional applications," in *UIST*, 2002.
[5] G. Paravati, A. Sanna, F. Lamberti, and C. Celozzi, "A reconfigurable multi-touch framework for teleoperation tasks," in *Emerging Technologies and Factory Automation*, 2011.
[6] K. Yamane, Y. Ariki, and J. Hodgins, "Animating non-humanoid characters with human motion data," *SCA*, 2010.
[7] L. Ikemoto, O. Arikan, and D. Forsyth, "Generalizing motion edits with gaussian processes," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1:1–1:12, 2009.
[8] J. Kofman, X. W., T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Trans. on Industrial Electronics*, vol. 52, no. 5, pp. 1206 – 1219, 2005.
[9] S. Chang, J. Kim, I. Kim, J. Borm, C. Lee, and J. Park, "Kist teleoperation system for humanoid robot," in *IROS*, 1999.
[10] O. Fukuda, T. Tsuji, M. Kaneko, and A. Otsuka, "A human-assisting manipulator teleoperated by emg signals and arm motions," *ICRA*, 2003.
[11] T. A. Rockall and A. W. Darzi, "Tele-manipulator robots in surgery," *British Journal of Surgery*, vol. 90, no. 6, 2003.
[12] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.
[13] C. Atkeson, "Using locally weighted regression for robot learning," in *ICRA*, 1991.
[14] M. Kaiser and R. Dillmann, "Building elementary robot skills from human demonstration," in *ICRA*, 1996.
[15] D. Grollman and O. Jenkins, "Sparse incremental learning for interactive robot control policy estimation," in *ICRA*, 2008.
[16] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," in *IEEE Trans. on Signal Processing*, 2004.
[17] S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. Dogar, A. Dragan, R. Knepper, T. Niemueller, K. Strabala, M. V. Weghe, and J. Ziegler, "Herb 2.0: A robot assistant in the home," *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, 2012.
[18] E. Lesaffre, "Superiority, equivalence, and non-inferiority trials." *Bull NYU Hosp Jt Dis*, vol. 66, no. 2, pp. 150–154, 2008.
[19] I. Scott, "Non-inferiority trials: determining whether alternative treatments are good enough." *Med J*, vol. 190, pp. 326–330, 2009.