Teaching by touching: an intuitive method for development of humanoid robot motions

Fabio Dalla Libera*, Takashi Minato†, Ian Fasel†, Hiroshi Ishiguro†‡, Emanuele Menegatti* and Enrico Pagello*
* Intelligent Autonomous Systems Laboratory, Department of Information Engineering (DEI),
Faculty of Engineering, University of Padua, Via Gradenigo 6/a, I-35131 Padova, Italy
† ERATO, Japan Science and Technology Agency,
Osaka University, Suita, Osaka, 565-0871, Japan
‡ Department of Adaptive Machine Systems, Osaka University, Suita, Osaka, 565-0871 Japan

***Abstract*— This paper investigates touching as a natural way for humans to communicate with robots. In particular we developed a system to edit motions of a small humanoid robot by touching its body parts. This interface has two purposes: it allows the user to develop robot motions in a very intuitive way, and it allows us to collect data useful for studying the characteristics of touching as a means of communication. Experimental results confirm the interface's ease of use for inexpert users, and analysis of the data collected during human-robot teaching episodes has yielded several useful insights.**

I. Introduction

In order for robots to become truly integrated into everyday life, it will be necessary for humans to be able to interact with them in a natural and intuitive way. This consideration has recently lead to many different studies in human-robot interaction with the aim of finding natural ways by which humans can communicate with robots (e.g., [1]–[4]).

Abstractly, we regard communication as a process by which a sender encodes a concept into a format suitable for transmission though a medium, and sends this information to a receiver, who then reconstructs (or decodes) it. We can divide human-to-human communication roughly into verbal (when concepts are encoded in the form of words) or non-verbal (see for instance [5]). Non-verbal communication can then be broken down further according to the transmission channel used, such as communication via vision, smell (a study on this communication medium is provided by [6]), or touch. There are a great variety of studies dealing with characteristics of visual communication, for example recognition of human gestures [2], analysis of how a robot's aspect affects human-robot interaction [7], and knowledge transfer through visual task recognition, as is in Kuniyoshi and Inoue's [8] work in which a robot observes humans performing a task and recognizes various actions, from which it constructs a high-level, abstract plan. However touch as a communication medium has received considerably less attention.

Touch is an important method of communication employed by humans, particularly in teaching. Even at the earliest ages, touching behaviors have been found to be a very important element of interactions between humans and preschoolers [9]. At older ages, touch is frequently used in the teaching of sports or dance [10], for instance by instructors correcting a learner's posture or motion. Touch is particularly appealing as an intuitive method for humans to teach robots, and has been employed to program robot arms, for example, by Voyles and Khosla in [3] and, more recently, by Grunwal, Schreiber, Albu-Schäffer and Hirzinger in [4].

In this paper, we investigate the effectiveness of touching as a mechanism for transferring knowledge about the body from a human to a small humanoid robot. Small humanoid robots are quite popular and are becoming increasingly available at relatively low cost. However teaching a new motion to a humanoid robot is currently a time consuming task, because the standard method is through the use of motion editors which require the user to set the position of each joint in each "keyframe" (as illustrated in [11]). Although other techniques, such as motion capture and retargetting [12], can be employed, these methods are still somewhat cumbersome, and require the human teacher to learn specialized techniques.

Our goal is to create a method by which humans can intuitively instruct a robot without any special training. We therefore have developed a method for humans to teach robot motions through an "observe and correct" cycle, similar to a human dance or sports instruction. In each teaching episode, the human teacher watches the robot perform a motion, observes what is wrong or could be improved, and touches the robot's body parts to instruct the robot how to modify the motion. For example, the teacher could watch a kicking motion and notice that the right leg should rise higher in a specific moment. S/he can then touch the leg from the back and push it upward to express how the motion should be modified. The robot then repeats the behavior with the modifications, and the cycle can be repeated several times until the movement is satisfactory to the teacher.

The teacher's touching actions are a method of encoding and transmitting their internal image of what the robot postures should be. To make communication successful, the robot must then interpret the meaning of these touches in terms of adjusted

body postures. However for the robot, this reconstruction process is not a trivial task. Not only can different touches have the same meaning – for instance, touching several different parts of the arm could all mean that the arm should move backwards – but similar touches could have different meanings depending on the context. For example if the robot is standing, touching the upper part of one leg could mean that that the leg should bend further backwards. However if the robot is squatting, the same touch could mean that the robot should move lower to the ground by bending its knees.

Rather than force the human user to learn an externally defined teaching protocol (determined by a programmer based on e.g., inverse kinematics) we instead take the approach that the mapping can be constructed online from examples provided by the user. While observing the robot perform a real task, the user chooses key moments to provide instruction. At these moments, the human touches parts of the robot, and the robot responds by moving its joints according to the learned mapping. If the robot's responses to touches are incorrect, the human can manually adjust its joints to teach the intended meaning using a separate interface (shown in Fig. 4). The robot then uses this as an example in which to update its internal mapping from touch to joint angle changes. As instruction progresses, the learned mapping continues to be refined until ultimately the human only needs to touch the robot and the robot properly adjusts its body.

In the remainder of this paper, we describe our interface and learning techniques in detail, and discuss how users can use this method to easily teach specific actions. We then perform analysis of the data collected during the touch teaching interactions, to help improve our understanding of how humans communicate via touch. We show that a simple linear model for mapping touch to joint angle changes cannot capture the dependence of touching behavior on context. We then use a tree-based feature selection method to get a sense of what variables are most important in explaining the changing meaning of human touch. Our discovery was that the most important variables to consider for context seem to be the positions of those joints which determine the overall orientation of each of the limbs. We finally conclude with a discussion of future work and suggest a possible way to further formalize the interaction between context, intention, touch, and robot body posture.

II. Implementation

The purpose of the interface is to let the user play a motion and modify it through touching, and provide information about the intended meaning of touches if needed. During the development of a motion, the user pushes the robot's body parts, and the robot tries to predict the intended joint angle changes based on previous examples of context, touches and joint modifications. Inference of intended joint changes due to touches is done using a k-nearest neighbor (k-NN) algorithm on a database containing previous examples of context, touch, and pose changes. If the human believes the robot does not yet have a good mapping, the user can directly set the joint position and add this example of context, touch, joint adjustments into the database of examples used by the k-NN algorithm, thereby improving the future ability to infer the touch intentions. This approach is similar to the critic method of [13], however their system required a fixed initial database of context, touch, and action examples, and the human critiques simply adjusted the relative weightings of these examples in their 1-NN inference engine.

Throughout the text, we use capital Roman letters to represent random variables, and lowercase Roman letters to represent specific values taken by those variables. Let touch information be represented by a vector $T = (T_1, ..., T_{n_T})$, where the value of each element represents the duration that each of the n_T tactile sensors distributed over the robot's surface were pushed. We encode context with the following random variables. Let:

- posture vector $P = (P_1, ..., P_{n_P})$ be the angles formed by each of the n_P joints,
- orientation vector $O = (O_1, O_2, O_3)$ be roll, pitch and yaw, respectively,
- velocity vector $V = (V_1, V_2, V_3)$ be the velocity at the center of gravity.

P is needed because the meaning of touches may depend on the posture, as in the previous example in which touching the lap means different things depending on if the robot is standing or squatting. Likewise, the meaning of touches may also depend on O, for instance whether the robot is standing or laying down. Finally, touches may also depend on the velocity vector V, especially if the robot is experiencing strong accelerations, for example if it is falling down. It is possible that the velocity of each single limb should also be included, however for the moment we felt this was excessive, and will put off investigating this and other features for future work.

To simplify notation, let $X = (T, P, O, V)$ be the concatenation of touch T, posture P, orientation O, and velocity V. Let $M = (M_1, ..., M_{n_M})$ be a vector of desired changes in the n_M joint angles (i.e., motor commands). Then our goal is to learn a policy function $F : X \rightarrow M$, which maps an input vector x to a set of joint angle changes m. Currently, we use a variation of the k-nearest neighbors algorithm, which, despite its simplicity, performs very well in many applications (e.g., [13], [14]). Aside from simplicity, an important reason we chose this algorithm was the ease with which additional training data can be incorporated.

Formally, let the tuple (x^i, m^i) represent the ith training example provided by the human, and let $\mathcal{S} = ((x^1, m^1), (x^2, m^2), ..., (x^{n_S}, m^{n_S}))$ be a set of n_S training examples. Sometimes we need to refer to specific elements of variables, so let t_s^i represent the value of element s of the touch sensors in the ith training example, and similarly for the other variables. Then, given a new input x', the system's output m' can be computed by a weighted sum of the joint modifications in $\mathcal{H}$, i.e.,

$$m' = \sum_{i=1}^{n_S} g(x', x^i) m^i \tag{1}$$

Fig. 1. Examples of considered ad discarded examples applying the described rule.

where the weight function $g(x', x^i)$ is based on the euclidean distance between test point x' and training point x^i.

A. Weighting schema

We want the weighting function g to output relatively larger values when the two inputs are close to each other, and relatively smaller values when the inputs are far away from each other. A simple choice for this function that has the desired properties is

$$g(x', x^i) = \frac{1}{1 + \|x', x^i\|} \tag{2}$$

where $\|\cdot, \cdot\|$ is euclidean distance. Since the units of the various input vector components are heterogeneous, it is important that each input vector component be normalized. This can be done by first dividing each element by its standard deviation in the example set, which is the same as replacing the distance function with a Mahalanobis distance using a diagonal covariance matrix.

Unfortunately, this technique does not give any "priority" to more important elements – for instance, the touch information does not get any more importance than the context features. This means that points with a very similar context (ex, similar posture) may dominate the determination of the output, irrespective of the touching pattern. This is exacerbated by the relatively high dimension of the input space and the limited number of example points. This can lead to very unintuitive behaviors, for instance if a user is focusing on a leg motion and therefore only provides examples involving a leg, then pushing on an arm will cause the leg to move, a surprising behavior indeed!

To solve this problem, we modify $g(x', x^i)$ to be zero if the set of activated (i.e., nonzero) touch sensors in t^i is not a subset of the active touch sensors in t', i.e. $g(x', x^i) = 0$ if

$$\prod_{\{s:t'_s=0\}} \left(1 - \delta(t^i_s)\right) = 0, \tag{3}$$

where the threshold function $\delta(u) = 1$ if $u > 0$ and 0 otherwise. Some examples are provided by Fig. 1.

With this modification, most of the counterintuitive behaviors are avoided, although it can make the mapping less general and it introduces some discontinuities when the set of pressed sensors varies. On the flip side, this modification yields a significant speed-up of the system – as much as 2000% in some tests. This makes it possible to calculate and display the predicted joint modifications in real-time, providing useful visual feedback to the user while they are touching the virtual sensors.

Another problem with this scheme is that, due to the symmetry of the distance function, it is not possible to distinguish whether a current input sensor has been pushed for a longer or shorter time than the nearby prototypes in the training set. This can lead to unintuitive behavior regarding the relationship between the duration of a touch and the magnitude of joint angle changes. As a simplistic example, suppose a particular sensor was active in only one training example, and it was pushed for 300 milliseconds, and this corresponded to a single motor joint change of 40 degrees. While a user might naturally expect that pushing for less time will cause a smaller change in that joint, while a longer press should produce a larger joint angle change, the result with the current scheme would be that any touch on that sensor with a duration different from 300ms would result in a smaller angle change. Fig. 2 illustrates this problem.

To overcome this counterintuitive behavior, we compute two factors, α_i and β_i, and then redefine g as

$$g(x', x^i) = \alpha_i \beta_i \prod_{\{s:t'_s=0\}} \left(1 - \delta(t^i_s)\right) \tag{4}$$

where

$$\alpha_i = \prod_{\{s:t^i_s>0\}} \frac{t'_s}{t^i_s} \tag{5}$$

is a value which increases linearly as the pushing time increases; the result is that increasing the pushing time of one sensor will only increase the weight of the examples in which that sensor was pushed, and it will not have an effect on the weights of other examples. The second factor β_i accounts for the context information, as well as for the touch sensors which are active in the input but are not active in the ith example. This is defined as

$$\beta_i = \frac{1}{1 + d_i} \tag{6}$$

where

$$d_i = \sqrt{\sum_{\{s:t^i_s=0\}} {t'_s}^2 + \|p' - p^i\|^2 + \|o' - o^i\|^2 + \|v' - v^i\|^2} \tag{7}$$

Essentially, d_i is a euclidean distance between x' and x^i, except ignoring the touch sensors which are nonzero in x^i (since they are used already in α_i). The specific choice of the form of β is admittedly arbitrary, and in future work we will investigate other types of weighting functions.

III. Experiment

For our experiments, we used a VStone[1] VisiON 4G, a humanoid robot with 22 degrees of freedom. Fig. 3(a) shows a

[1]http://www.vstone.co.jp

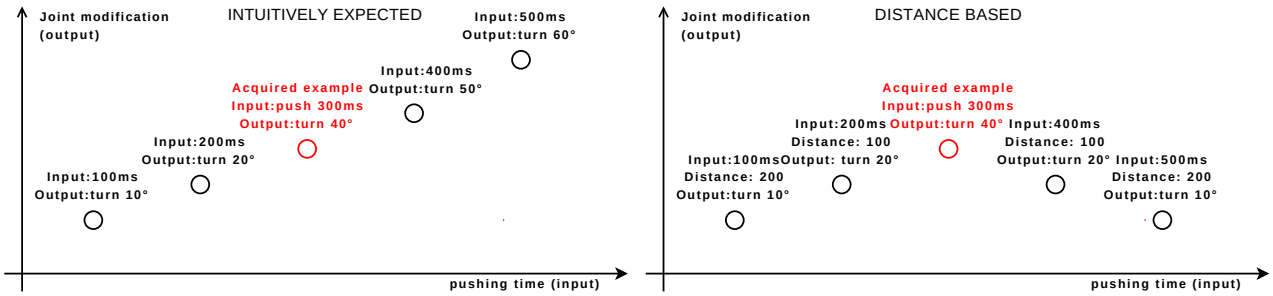
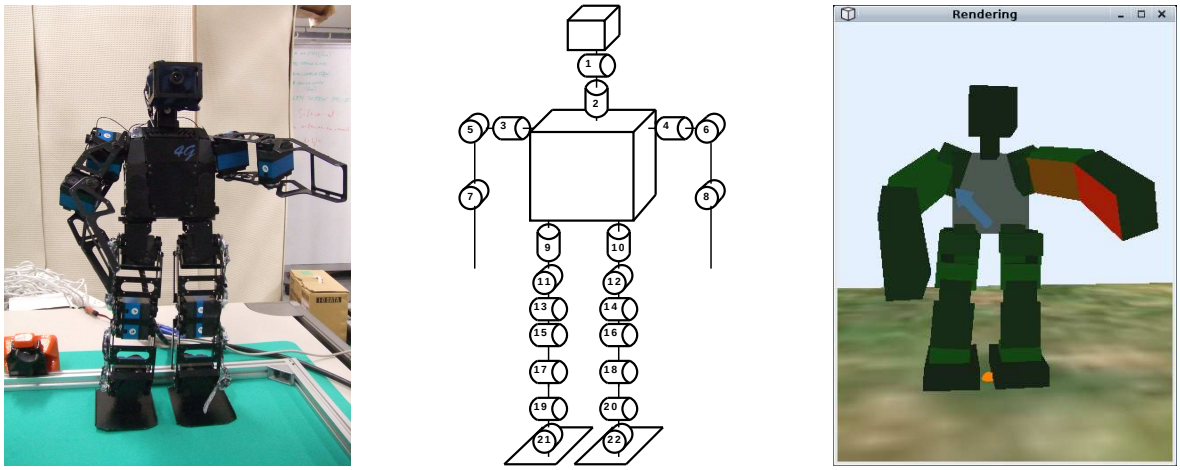


Fig. 2. Expected behavior versus the behavior obtained scaling the output by a decreasing function of the distance



(a) (b) (c)

Fig. 3. (a) A photo of the VStone VisiON 4G humanoid robot, (b) a schematic of its joints, and (c) the 3D rendered model, where the left upper arm and forearm have just been pushed. The 3D model display also shows the projection of the center of gravity onto the ground (represented by an orange sphere) and its velocity (a blue arrow) which can be useful to expert users in motion development.

photo of the robot, and a diagram depicting the configuration of the joints is given in Fig. 3(b).

It is impractical to use touch sensors directly on the robot's body for several reasons. This robot, and others like it, are typically quite small (often under 50cm/20inches tall), and use servomotors with an internal PID controller. With such devices, it is not possible to the detect the force applied by the user as might be done in e.g., larger pneumatic-actuator robots, in which it is possible to measure the error between the target position and the actual position of the actuators ([15]). The small size of these humanoids also makes it difficult to place and wire touch sensors over the entire body. Another difficulty with attempting to directly use touch for teaching these small robots is that the robot motions are often quite fast, so real-time interaction might be impossible for a human (especially if the robot is flying through the air as in a jumping motion).

To overcome these issues, we developed a system which combines the real-world robot actions with a virtual touch-screen driven interface. In this system:

1) A motion is performed by the physical robot, and the position of the robot body is recorded with a real-time-motion capture system. The use of motion capture on the real robot helps prevent any simulator-reality gap.
2) A computer interface allows to the human to watch a virtual 3D reconstruction of the recorded motion performed by the robot. The human can pause, rewind, and step through frames at their leisure.
3) The user chooses an instant where the posture of the robot should be modified, and playback is frozen at that point.
4) The human touches the robot model's body parts on a touch screen to modify the robot's posture.

Currently we use a touch screen but other devices, such as a haptic joystick, or simply a mouse, could also be used. When the posture in one moment is modified, it becomes a new keyframe, and the motion in the surrounding frames are then altered via interpolation. In the current implementation a simple linear interpolation is used, but the software has been designed to easily permit the use of more sophisticated interpolations, for instance the method used by [16].

While the robot is performing a motion, the joint positions

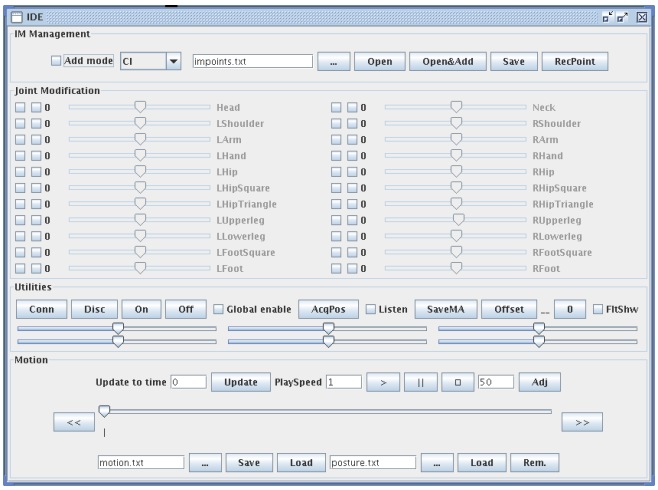


Fig. 4. A snapshot of the commands window of the developed interface

are acquired using the potentiometers present in each of the servomotors, while the overall orientation of the robot is captured using a Motion Analysis Corp. capture system[2]. The on-screen playback (implemented using Java 3D) displays the robot's limbs as parallelepipeds with size and joint positions proportional to the limb size and joint positions of the real hardware. Each parallelepiped's face simulates a touch sensor. Because the touch screen currently only tracks a single point, and discards pressure information, the user is allowed to touch various parts of the 3D-model in one keyframe, and the duration of each touch is considered to be the pushing intensity. As the user touches the robot, the parts being pushed become more and more red (see Fig. 3 c), so the user knows what input the robot is receiving.

If the system fails to predict the desired modification, which can be immediately seen by the robot's response, the user can manually correct the robot's joints. To do so, the interface allows the human to independently switch off any of the motors, and the human can then move the the limbs of the physical robot into the desired position. To fine tune the various angles, the interface provides one slider for each of the servomotors, as shown in fig. 4. The system can then acquire the robot's new joints angles, and then stores the context, touch, and joint angle changes as a new training example.

We used the interface to teach two motions: jumping (with the help of a rubber band pulling the robot up since the servo torque is not sufficient for liftoff) and walking. Let the set of examples collected during these teaching episodes be denoted by $JUMP$ and $WALK$, respectively. Fig. 5 shows an image sequence of the learned jumping motion, which was taught in just seventeen minutes. Teaching the same motion using a traditional motion editor took more than forty minutes.

[2]for details see `http://www.motionanalysis.com/applications/movement/neuro/eaglesystem.html`

IV. Data Analysis

We performed two kinds of analysis, both of them aimed mainly at understanding the role of context. First, we wanted to see if the mapping from touch to change in joint angles could be learned simply with linear regression. This is useful both as a baseline to compare with other methods, and to help develop some intuitions about what makes the problem difficult, for instance understanding what kinds of touches change their meaning due to context. The second analysis was a clustering / feature-selection analysis, in which we tried to understand which pose, orientation, and velocity variables are most closely associated with changing touch behaviors.

For the first analysis, using the definitions of the random variables from Section II (organized as row vectors), let

$$M = X\mathbf{A} + \epsilon, \tag{8}$$

be a linear model of input to output, where $\mathbf{A}$ is a matrix whose number of rows is the same as the number of elements of X and number of columns is the number of joint angles n_M, and ϵ represents Gaussian noise with zero mean and spherical covariance. Given a set of training data, organized into a matrix of inputs $\mathbf{X}$ and outputs $\mathbf{M}$, the matrix $\mathbf{A}$ minimizing the squared error can be found by ridge regression [17], i.e.,

$$\hat{\mathbf{A}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{M} \tag{9}$$

where the T superscript indicates the transpose operation, $\mathbf{I}$ is the identity matrix, and α is a small number which can be interpreted as an estimate of the standard deviation of the Gaussian noise. The value of α is chosen heuristically, by manually finding a value that gives low cross validation error. We fixed $\alpha = 0.1$ for all experiments, then found $\hat{\mathbf{A}}$ using the entire training dataset for each experiment.

Looking at the magnitudes of the entries in $\hat{\mathbf{A}}$ provides insight into the importance of each of the input features, while the sign helps us to understand which features produce similar effects and which produce opposite effects. Fig. 6 gives a visual representation of $\hat{\mathbf{A}}$ superimposed on the model of the robot. In this figure, the touch sensors of the robot are colored according to their corresponding row in $\hat{\mathbf{A}}$ for two different joints (each corresponding to a different column in $\hat{\mathbf{A}}$). For example in Figs. 6c and 6d, corresponding to joint 7, each face is colored green if the corresponding entry in column 7 of $\hat{\mathbf{A}}$ is positive, and red if it is negative, while the color intensity represents the magnitude of the value. In general the values follow intuition fairly well – for instance, for determining the arm orientation changes, the sensors with high coefficients are mainly the ones on the arm.

In Table I, we show the prediction error when the matrix learned from one of the training sets (or from the combined dataset), is tested for its ability to predict itself, the other dataset, or the combined dataset. Each entry shows the average euclidean distance. For comparison, we compute the same values for the k-NN algorithm described in Section II. We can see that although the linear regression model is sometimes able to make very accurate predictions, at other times it fails

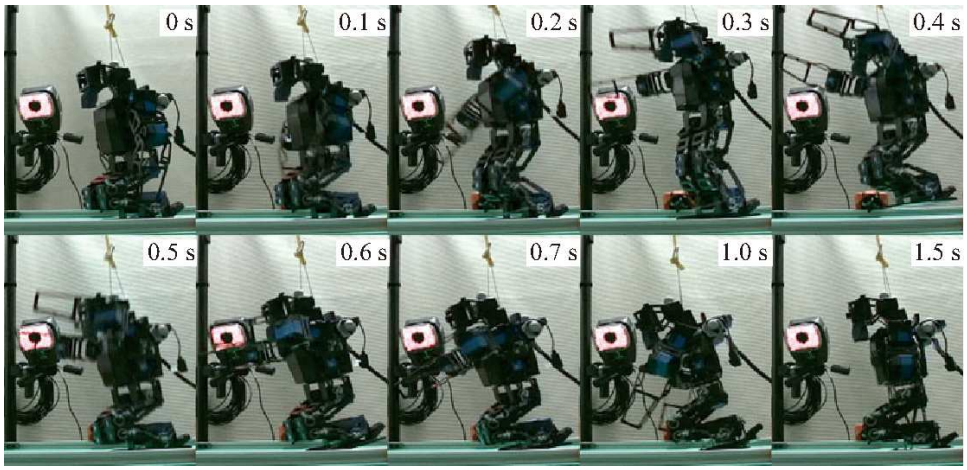
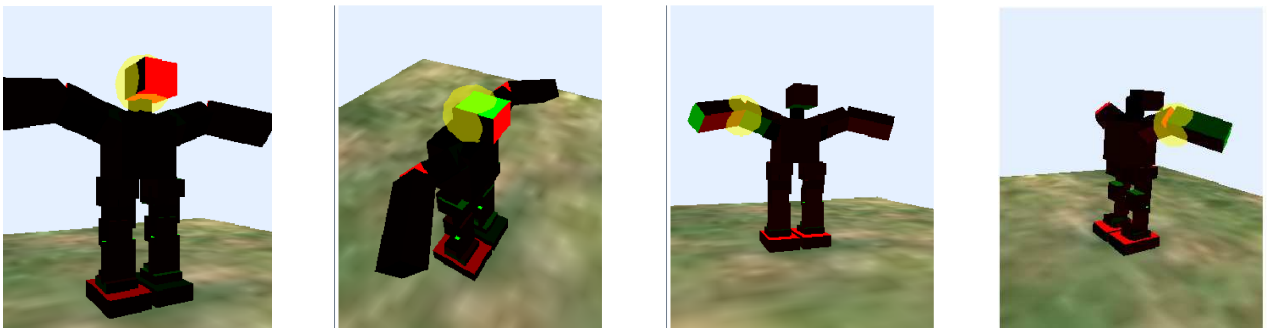


Fig. 5. An image sequence of the developed jumping motion



(a) (b) (c) (d)

Fig. 6. The importance of each of the sensors in determining the variation of the head orientation ((a) and (b), joint 1 of fig. 3(b)) and the variation of the elbow joint angle((c) and (d), joint 7 of fig. 3). The various sensors are coloured red or green depending on the sign of the relative coefficient, while its absolute value is represented by the intensity of colour. A yellow sphere highlights the position of the joint which depends on the coefficients used to colour the sensors.

TABLE I
AVERAGE RMSD ERROR BY LINEAR REGRESSION AND K-NN

Training dataset	Test dataset	Lin. Regr.	k-NN
JUMP	JUMP	0.2846	0.1985
JUMP	WALK	13.7177	0.5938
JUMP	COMBINED	2.0567	1.0198
WALK	JUMP	0.0314	0.0588
WALK	WALK	0.3522	0.2028
WALK	COMBINED	0.3626	0.2223
COMBINED	JUMP	0.3219	0.1425
COMBINED	WALK	3.7272	0.3006
COMBINED	COMBINED	1.5462	0.7778

badly. Meanwhile, our k-NN algorithm performs consistently well, even when predicting the touch mapping for different action sequences.

A. *Context feature selection*

We hypothesize that the reason the nonlinear technique works better than the linear regression model is because the human produces touches differently based on the robot context variables. We illustrate this idea in Fig. 7. In this model, the human has an internal belief about how joints should be modified in order to improve the task performance. Meanwhile, the human also evaluates some features of the context to determine how to best communicate this desire to the robot. The pattern of touches is thus jointly determined by the desired changes and the current robot context. When the human simultaneously touches and directly manipulates the robot joints, we can observe all of the nodes in the lower part of the graph – the context, the touches, and the intended joint angle changes – while the human's intention and perception of the context remain hidden. However, when the human is not providing direct joint manipulation, the corresponding node becomes hidden, and the goal of the robot is to infer the values at that node from the observed touch and context data.

Because we believe the touching behavior is jointly deter-

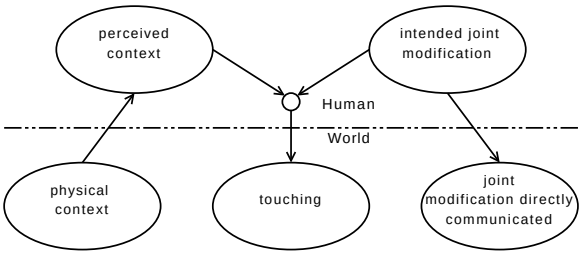


Fig. 7. A schema resuming the various concepts.

mined by the desired joint changes and the inferred context, we wanted to see if it is possible to determine a few context variables that can predict when the touching behavior will change – i.e., when the mapping from touches to joint angle changes becomes non-linear. To do this, we used a tree-based technique, using Quinlan's C4.5 algorithm [18]. The most telling indicator of contextual influence is when the same touch causes a joint to be adjusted in a different direction. Therefore, we created a simplified problem of discriminating direction of joint angle change given touch and context. For each training example x^i we create a target variable $q^i = (q^i_1, ..., q^i_{n_P})$ based on p^i (the pose component of x^i), where each element is set to $+1$ if the corresponding element in p^i is larger than some threshold τ, -1 if it is smaller than $-\tau$, and 0 otherwise. In practice we set τ to a value of 5 degrees. Note that although in principle this could result as many unique target values as training points, in practice of the 238 collected examples, there were only 97 unique values of q. We then ran the Q4.5 algorithm, with pruning, on each of the datasets (JUMP, WALK, and the combination), which tries to predict the direction of joint angle changes Q from the touch and context variables X. We then examined where each context variable appeared in the trees, to see if any of these consistently appeared high in the tree (suggesting they have a great influence in determining the direction of change). Table II shows the level at which the various context features appear (notice that one feature can appear multiple times, along different paths of the tree).

The most interesting observation from Table II is that for all data sets, the position of the joints near the trunk of the body, which determine the global position of the limbs, are used as a discriminating attribute in high (near the root) levels of the three. Although we did not predict this beforehand, this makes intuitive sense, suggesting that the meaning of the touch depends primarily on the overall positions of the limbs.

Also somewhat surprising from this analysis is the fact that the robot orientation and velocity do not appear very high in any of the trees. We hypothesize that this might not be the case were we teaching more dynamical motions, and so in future work we will investigate this hypothesis more deeply.

TABLE II

LEVELS IN THE DECISION TREE OF THE FEATURES

Feature \ Dataset	JUMP	WALK	COMBINED
joint 1	2		2
joint 2	17,21,22	9	19,23
joint 3	3	3	3
joint 4		5,11	
joint 6		2	
joint 7	20		22
joint 8	5,8,19		5,10
joint 9	1	3	1
joint 11		10	
joint 12	0	1	0
joint 14		3	
joint 15		2,8,10	
joint 16	5	3,4	7
heading		0,2	

Levels at which the various features appear constructing the three with each of the three matrices as training data. Only the features which appear at least in one of the trees had been reported inthe table. The joint numbers are the ones reported in fig. 3(b). The entry "heading", is part of the heading, attitude and bank triple used to define the orientation.

V. CONCLUSION AND FUTURE WORK

We have developed an interface for teaching robots through touching, which allows the user to continuously refine the meaning of their touches by directly manipulating the limbs of the robot. The interface we have developed is readily usable by inexpert users, and allows people to teach robots new behaviors much more quickly than through more traditional motion editors. We have hypothesized that one reason a linear model is not sufficient to learn the mapping from context and touch to changes in joint angles is because context and touch interact in a nonlinear way. We applied the C4.5 algorithm as a first step for gaining insights about which features of the physical world are related to how humans express their intentions through touching. This can be interpreted as attempting to learn the mapping from the physical context to the perceived context, using the touch and joint angle information to infer the different categories of perceived context.

Our k-NN algorithm does not directly represent the context, touch, and human intention as separate variables. However it is possible that we could divide the context into a few basic categories (e.g., squatting, standing, etc.), perhaps using the features discovered from our tree-based analysis, and then learn a separate kNN model, or perhaps even a linear model, for each context category. In future work, we intend to model each of the nodes shown in Fig. 7 as either discrete or Gaussian (or mixtures of Gaussian) random variables, and use more sophisticated Bayesian techniques to infer their parameters given training data. In this way we can jointly learn the set of context categories, the variables that are needed to infer the context, and the mapping from touch to joint modifications given context.

An important future goal for this work is to apply similar methodologies to bigger robots with more sensors and actuators. In such a scenario, the virtualization of the touch sensors will not be necessary, except perhaps for actions that are too

fast for the human to manipulate in real-time. This amplifies the advantage of our approach, since complex robots with many sensors and actuators are even more difficult to program using traditional motion editors. Moreover, the use of real sensors may make it possible to discriminate between different kinds of the touch, for example differentiating between long soft touching patterns and short, which can increase the range of possible human instructions. From a technical standpoint, we have written our code with portability in mind, in order to make migration to new platforms such as human-size androids relatively painless.

As we continue to improve this interface, we will be able to perform further studies in the analysis of touching as an intuitive way for users to interact with robots. We are particularly interested in further investigating the importance of various context features, and how they influence the intended message conveyed through touching communication. Importantly, we believe it will be important to extend the context to include timing related features, such us the elapsed time between the maximum/null robot's acceleration and the pushing time. We expect further development of this system to yield interesting new insights into the human touching behavior, as well as a more effective method for teaching robots.

REFERENCES

[1] A. Stoica. Humanoids for lunar and planetary surface operations. NASA Jet Propulsion Laboratory. Pasadena, CA. [Online]. Available: http://hw.jpl.nasa.gov/humanoid/assets/Documents/\\STAIF_stoica_Final%_Camera_Ready.pdf

[2] C. Lee and Y. Xu, "Online, interactive learning of gestures for human/robot interfaces," in *IEEE International Conference on Robotics and Automation*, Minneapolis, MN, 1996, pp. 2982–2987.

[3] R. Voyles and P. Khosla, "Tactile gestures for human/robot interaction," in *Proc. of IEEE/RSJ Conf. on Intellingent Robots and Systems*, Vol.3, 1995, pp. 7–13.

[4] G. Grunwald, G.Schreiber, A. Albu-Schaffer, and G. Hirzinger, "Touch: The direct type of human interaction with a redundant service robot," in *Proc. of the IEEE Int. Workshop on Robot and Human Interactive Communication*, Bordeax/Paris,France, 2001.

[5] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone., "Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents," in *SIGGRAPH*, 1994, pp. 413–420.

[6] A. Bodnar, R. Corbett, and D. Nekrasovski, "Aroma: Ambient awareness through olfaction in a messaging application," in *Proceedings of the International Conference on Multimodal Interfaces*, PA, 2004.

[7] T. Minato, M. Shimada, S. Itakura, K. Lee, and H. Ishiguro, "Evaluating the human likeness of an android by comparing gaze behaviors elicited by the android and a person," in *Advanced Robotics*, Vol.20 No.10, 2006, pp. 1147–1163.

[8] M. Kuniyoshi and I. Inoue, "Learning by watching: extracting reusable knowledge from visual observation of human performance," in *Proc. of IEEE Transactions on Robotics and Automation*, Vol.10 No.6, 1994, pp. 799–822.

[9] J. R. Movellan, F. Tanaka, I. R. Fasel, C. Taylor, P. Ruvolo, and M. Eckhardt, "The RUBI project: a progress report." in *HRI*, C. Breazeal, A. C. Schultz, T. Fong, and S. B. Kiesler, Eds. ACM, 2007, pp. 333–339.

[10] K. Kosuge, T. Hayashi, Y. Hirata, and R. Tobiyama, "Dance partner robot -ms dancer-," in *Proc. of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas,Nevada, 2003.

[11] T. Wama, M. Higuchi, H. Sakamoto, and R. Nakatsu, "Realization of tai-chi motion using a humanoid robot," in *Proc. of The 14th International Conference on Artificial Reality and Telexistence*, 2004, pp. 71–74.

[12] A. Nakazawa, S. Nakaoka, K. Ikeuchi, and K. Yokoi, "Imitating humand dance motions through motion structure analysis," in *Proc. of the 2002 IEEE-RSJ Intl. Conference on Intelligent Robots and Systems*, Lausanne,Switzerland, 2002.

[13] B. Argall, B. Browning, and M. M. Veloso, "Learning by demonstration with critique from a human teacher." in *HRI*, C. Breazeal, A. C. Schultz, T. Fong, and S. B. Kiesler, Eds. ACM, 2007, pp. 57–64.

[14] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, "Machine learning, neural and statistical classification," in *Ellis Horwood*, London, 1994.

[15] B. McDonell and J. Bobrow, "Modeling identification and control of a pneumatic actuated, force controllable robot," in *IEEE Trans. Robot Automat.*, Vol 14, 1998, pp. 732–742.

[16] N. Mayer, J. Boedecker, K. Masui, M. Ogino, and M. Asada, "HMDP:a new protocol for motion pattern generation towards behavior abstraction." in *RoboCup Comp. Symp.*, Atlanta, 2007.

[17] A. Hoerl and R. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, pp. 55–67, 1970.

[18] J. Quinlan, *C4.5 Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[19] C. Breazeal, A. C. Schultz, T. Fong, and S. B. Kiesler, Eds., *Proceedings of the Second ACM SIGCHI/SIGART Conference on Human-Robot Interaction, HRI 2007, Arlington, Virginia, USA, March 10-12, 2007*. ACM, 2007.