Teaching by Touching: an Intuitive Method for Development of Humanoid Robot Motions

Fabio Dalla Libera (Univ. of Padua), *Takashi Minato (JST), Ian Fasel (Univ. of Texas), Hiroshi Ishiguro (Osaka Univ.), Emanuele Menegatti, and Enrico Pagello (Univ. of Padua)

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.

Key Words: Human-robot interaction, intention understanding, touch communication, motion development

1. 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.

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 [1]. At older ages, touch is frequently used in the teaching of sports or dance [2], 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 [3] and, more recently, by Grunwal et al. [4].

In this study, 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." Although other techniques, such as motion capture and retargetting [5], can be employed, these methods still require the human teacher to learn specialized techniques.

Our goal is to create a method by which humans can intuitively edit a robot motion 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 refine the motion. 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 action is 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.

In order for the robot to understand the meaning of the teacher's touch, we take the approach that the mapping can be constructed online from examples provided by the teacher. While observing the robot performs a real task, the teacher chooses key moments to provide instruction. The teacher touches parts of the robot, and the robot responds by moving its joints according to the learned mapping. It is worth to stress that this is different from what is usually done with robot arms employed, for instance, in painting and welding. In those applications, in fact, the operator pushes the robot in desired position, and the arms moves passively following the applied forces. In our approach the robot responds in a more active manner, possibly moving joints which would not be moved just applying force on the pushed sensor locations. The idea here introduced is that the system should interpret the meaning of the touching, and not require the user to bring each link in the desired position for each keyframe as is usually done with the industrial robots.

If the robot's responses to touches are incorrect, the teacher can manually adjust its joints to teach the intended meaning using a separate interface. This simpler, lower level way of communicating motions allows to handle the failures of the system in the interpretation of the instruction meaning. The robot then uses this information 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.

2. Implementation

The purpose of the interface is to let the user (teacher) 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.

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 lying 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.

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 function $F : X \rightarrow M$ ("touch interpreter"), which maps an input vector x to a set of joint angle changes m. Currently, we use a variation of the k-NN algorithm, which, despite its simplicity, performs very well in many applications. 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^i_s 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.,

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

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

2·1 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

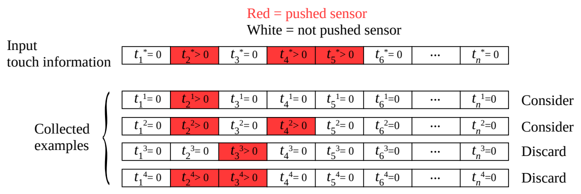
$$g(x', x^i) = 1/(1 + \|x', x^i\|), \qquad (2)$$

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.

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

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



Red = pushed sensor
White = not pushed sensor
Input touch information
$t_1^*=0$ $t_2^*>0$ $t_3^*=0$ $t_4^*>0$ $t_5^*>0$ $t_6^*=0$ ··· $t_n^*=0$
Collected examples
$t_1^1=0$ $t_2^1>0$ $t_3^1=0$ $t_4^1=0$ $t_5^1=0$ $t_6^1=0$ ··· $t_n^1=0$ Consider
$t_1^2=0$ $t_2^2>0$ $t_3^2=0$ $t_4^2>0$ $t_5^2=0$ $t_6^2=0$ ··· $t_n^2=0$ Consider
$t_1^3=0$ $t_2^3=0$ $t_3^3>0$ $t_4^3=0$ $t_5^3=0$ $t_6^3=0$ ··· $t_n^3=0$ Discard
$t_1^4=0$ $t_2^4>0$ $t_3^4>0$ $t_4^4=0$ $t_5^4=0$ $t_6^4=0$ ··· $t_n^4=0$ Discard

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

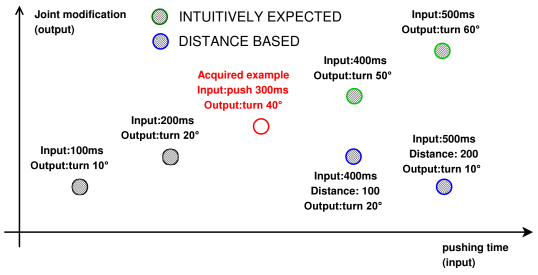


Fig.2 Expected behavior versus the behavior obtained scaling the output by a decreasing function of the distance.

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

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. Although 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 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

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

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

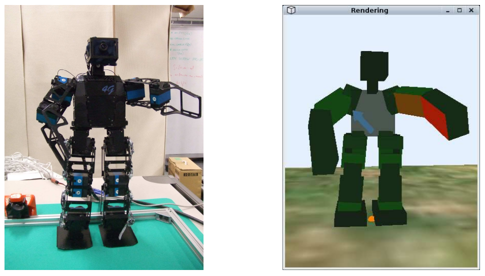


Fig.3 (a) Humanoid Robot VisiON 4G. (b) The 3D rendered model, where the left upper arm and forearm have just been pushed. It also shows the projection of the center of gravity onto the ground (an orange sphere) and its velocity (a blue arrow).

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

$$\beta_i = 1/(1 + d_i), \tag{6}$$

where

$$d_i^2 = \sum {t'_s}^2 + \|p' - p^i\|^2 + \|o' - o^i\|^2 + \|v' - v^i\|^2. \tag{7}$$

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).

3. Experiment

We used VisiON 4G, a humanoid robot with 22 degrees of freedom (Fig. 3(a)). It is impractical to use touch sensors directly on the robot's body for several reasons. The small size of these humanoids 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 difficult for a human.

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 motion capture system.
2. A computer interface allows to the user to watch a virtual 3D reconstruction of the recorded motion performed by the robot. The user 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 user touches the robot model's body parts on a touch screen to modify the robot's posture.

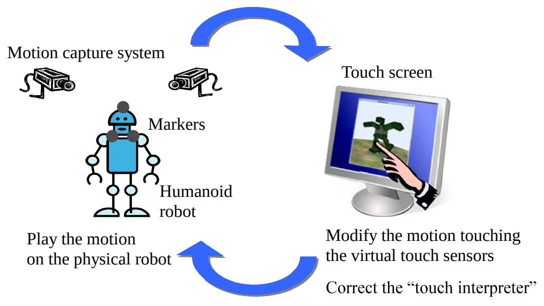

Motion capture system
Touch screen
Markers
Humanoid robot
Play the motion on the physical robot
Modify the motion touching the virtual touch sensors
Correct the "touch interpreter"

Fig.4 Phases of the motion development in the experiment.

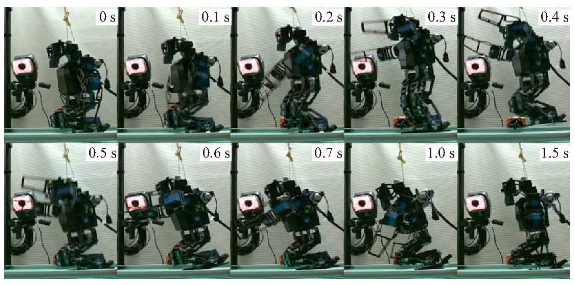


Fig.5 An image sequence of the developed jumping motion.

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 is then altered via interpolation. In the current implementation a simple linear interpolation is used.

The joint positions are acquired using the potentiometers and the overall orientation of the robot is captured using a motion capture system. The onscreen playback displays the robot's links as parallelepipeds with size and joint positions proportional to the link 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 darker red (see Fig. 3(b)).

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 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. The system can then acquire the robot's new joint angles, and then stores the context, touch, and joint angle changes as a new training example. Fig. 4 depicts the motion development process.

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. At the beginning, the robot always failed to interpret the touch meaning and the user needed to correct the robot's interpretation. The frequency of the correction gradually decreased as the robot acquired the examples. We could finally develop both motions. 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. More comparisons on the teaching time required in the future work.

4. 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. Through the experiment, it is shown that the robot is able to understand the touch meaning from the touch sensor information and contextual information (robot's posture, orientation, and velocity). The result will be important when the robot needs to construct the touch interpreter in unsupervised manner. We are now investigating which information is important for understanding touch meaning. We think that "Teaching by touching" will provide us not only an useful interface to develop a robot motion but also a research framework to study how a robot or a person understands other's intention.

[1] J. R. Movellan, F. Tanaka, I. R. Fasel, C. Taylor, P. Ruvolo, and M. Eckhardt. The RUBI project: a progress report. In *Proc. of the ACM/IEEE international conference on Human-robot interaction*, pp. 333–339, 2007.

[2] K. Kosuge, T. Hayashi, Y. Hirata, and R. Tobiyama. Dance partner robot -ms dancer-. In *Proc. of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3459–3464, 2003.

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

[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 International Workshop on Robot and Human Interactive Communication*, 2001.

[5] A. Nakazawa, S. Nakaoka, K. Ikeuchi, and K. Yokoi. Imitating human dance motions through motion structure analysis. In *Proc. of the IEEE/RSJ International. Conference on Intelligent Robots and Systems*, pp. 2539–2544, 2002.