Programming robot motions by touching

Fabio DallaLibera
DEI, University of Padua
Via Gradenigo 6/a I-35131
Padova, Italy
fabio.dallalibera@dei.unipd.it

Takashi Minato
ERATO, Japan Science and Technology Agency
Osaka University
Suita, Osaka, 565-0871, Japan
minato@jeap.org

Hiroshi Ishiguro
Department of Adaptive Machine Systems
Osaka University
Suita, Osaka, 565-0871 Japan
ishiguro@ams.eng.osaka-u.ac.jp

Enrico Pagello
DEI, University of Padua
Via Gradenigo 6/a, I-35131
Padova, Italy
epv@dei.unipd.it

Emanuele Menegatti
DEI, University of Padua
Via Gradenigo 6/a, I-35131
Padova, Italy
emg@dei.unipd.it

ABSTRACT

Small humanoid robots are becoming a more and more popular device for entertainment. While the literature shows many advanced techniques for the development of humanoid robot motions, for example motion retargeting, most of the commercially available products only provide a low level interface where each motion is described in terms of keyframes for which each joint angle is defined. In this paper we suggest to employ tactile interaction as an intuitive way for users to communicate with robots. The interface we developed serves two purposes: it allows the user to develop robot motions in a very natural way, and it allows us to collect data useful for studying the characteristics of touching as a means of communication. A supervised learning algorithm suited to extract the meaning of touch patterns is presented, and preliminary experiments that validate the feasibility of the approach are presented.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Haptic I/O

General Terms

Humanoid Robot, tactile interaction

1. INTRODUCTION

In the domestic robotics field humanoid robots are becoming a more and more easily available entertainment device on the market, at a lower and lower cost. Among the numerous commercial products we can cite for instance VStone's Robovie-X, Kondo kagaku's KHR-HV series or Hitech's Robonova (see `http://www.vstone.co.jp`, `www.kondo-robot.com` and `http://www.hitecrobotics.com/` respectively). These robots usually consist of rigid link body parts actuated by servomotors, whose target position is set by a control board (often by a Pulse Width Modulation signal). Most interfaces for their programming appear as the one in Fig. 1.

The target position of each of the servomotors is specified by the user using a slider [7] for some time instants, normally called keyframes. The robot control board then interpolates the keyframes to generate the motion. The motion process therefore requires the user to identify, for each posture, which are the parts of the robot which should be moved, realize which are the joints, along the kinematic chain, that cause the desired movements, and determine, for each of the joints, the right rotation direction and the appropriate rotation amount. Although other techniques, such as motion capture and retargeting [4], can be employed, these methods are still cumbersome, and require the use of expensive devices.

Touch is an intuitive method of communication employed in human-machine interaction [5], as the success on the market of newly introduced touch screen based cellular phones and music players seems to confirm. Human-human interaction, particularly teaching also benefits from tactile interaction. For instance touch is frequently used by sports coaches or dance instructors [6] to correct a learner's posture or motion. Tactile interaction therefore appears particularly appealing as an intuitive method for humans to teach robots, and in fact the literature shows how it can be used to program robot arms [2].

Our goal is to validate the idea of employing touch as a way for inexperienced users to program robot motions intuitively. As a concrete example, suppose a user just bought a small humanoid robot and wants to develop a walking motion. The systems available on the market requires her or him to think at the motion in terms of keyframes and for each of this key-postures the angles of each robot joint must be specified by a moving slider. This process is not straightforward, since, for instance, the user must identify which is the joint that moves the robot parts in the desired direction. Conversely, we can imagine that for novice users

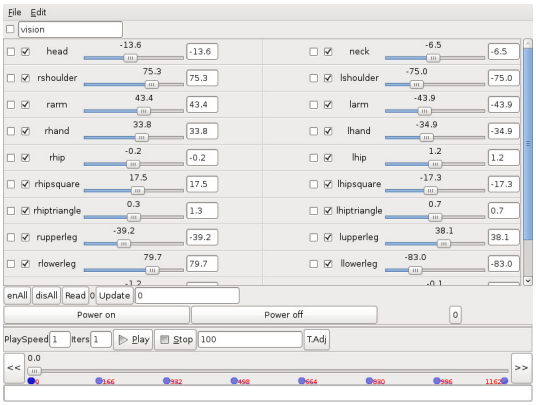

Figure 1: A classical slider based interface.

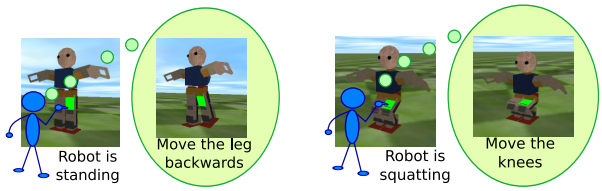

Figure 2: An example of the context dependence of the touch meaning. The user presses the same sensor, but due to the different robot posture the desired posture modifications (bend the leg and bend the knees, respectively) differs.

directly touching the robot's leg would be quite intuitive, as in general with kinesthetic demonstration [3].

Essentially the teachers' touching is a method of transmitting their internal image of what the robot postures should be. To make communication successful, the robot must then interpret these touches in terms of adjusted body postures. However, for the robot this reconstruction process is not straightforward since 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 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 (see Fig. 2).

Furthermore, the style and method of touching could be in part or totally user-dependent. Defining a fixed protocol and forcing the user to employ it would allow to solve these ambiguity problems. However, we believe that by making the robot's instruction interpretation adaptive users will be able to touch the robot more naturally and therefore develop motions with a very low mental effort.

Our interface allows users to switch between two modalities. In the "motion-development" phase the users press touch sensors and the robots moves according to its interpretation of the touch pattern. When users feel that the robot interpretation does not reflect their intention, they just switch to the "touch-meaning-teaching" phase and provide, by other means (direct robot manipulation or classical, slider based interface) the posture modification they intended by the applied touch. In this way the (supervised learning) algorithm for touch interpretation can be refined more and more online, i.e. during motion development. This approach presents an interesting secondary advantage: the collected training examples, consisting of a touch pattern and the corresponding desired joint modification, can be studied afterwards to help improve our understanding of how humans communicate via touch.

The development cycle and the learning algorithm will be described in detail in section 2. Performed proof-of-concept experiments are briefly reported in section 3. Section 4 concludes presenting a model for supervised tactile interaction and describes future works.

2. INTERFACE

As reported in the introduction our system allows to develop motions by touching. In particular the development cycle alternates execution and editing of the motion. The motion is played on the real, physical robot and the actual evolution of the motion is recorded. The user then watches the executed movement by a 3D representation of the robot, selects the instant in time where the motion should be modified and touches the robot to modify the posture in that time instant. Employing a virtual representation of the robot instead of the robot itself for the motion editing allows to use this approach for all kind of robot, even for those not equipped with touch sensors, as most of the humanoids available on the market. This representation does not need to simulate the dynamics and therefore can be a rough approximation of the robot. In fact our interface simplifies the robot links by parallelepipeds, and simulates a touch sensor on each of the faces. Expressly the sensors, which are assumed to be binary, can be pushed by clicking them with the mouse, and the clicking time of each sensor is recorded and given as input to the touch interpreter algorithm. Assuming binary sensors gives the system high robustness, and tests performed on a robot equipped with touch sensors showed that real touch sensors can be employed seamlessly. The touch interpreter then uses the examples of touch pattern and corresponding joint change to infer the desired joint modification. As stated in the introduction the mapping is context dependent, so to be more precise the algorithm realizes a mapping from a touch pattern and a context to a joint modification. Currently the context consists of:

- the angle of each of the joints, as the meaning of touches may depend on the posture (Fig. 2);
- the orientation (roll, pitch and yaw) of the robot, since the meaning can change, for instance, whether the robot is standing or lying down;
- the velocity vector of the center of gravity, since, for instance, the meaning could be different if the robot is falling down.

Distance weighted nearest neighbor algorithm with $k = \infty$ was employed to realize the mapping. More concretely each example provided by the user consists of an input (touch pattern and context) I_i and an associated intended joint modification vector M_i. Given an input I_*, the system output

vector M_* can be obtained by weighting the joint modifications present in the collected examples M_i, with weights ω_i calculated employing the distance (in the high dimensional space) between the system input I_* and each example coordinates I_i. Expressly, denoting by E the number of collected example $M_* = \sum_{i=1}^{E} \omega_i M_i$

Employing a decreasing function of the Euclidean distance for the weights was shown to give poor estimations of the desired joint modification, so we devised a specific metric. First of all, touch information (pushing time of each of the sensors) should be prioritized over the context. As a trivial example, if the human operator designed an arm motion and therefore only provided examples involving the arm, then when he/she will push sensors on the leg the arm will move depending on the stored closest context, while in such cases of no available knowledge it would be intuitive not to apply any posture modification. To avoid these situations our algorithm sets $\omega_i = 0$ for the examples where a sensor pressed in the example is not pressed in the input I_*. The second problem arises because every distance is a symmetric function. Suppose to have just one training example, where a sensor was pushed for τ milliseconds, and this corresponded to a single motor joint change. A user might expect that pushing for less time will cause a smaller change in that joint, while a longer press should produce a larger angle variation. Conversely the system behavior with a distance based weighting would be that any touch on that sensor with duration different from τ, both longer and shorter, would result in a smaller angle change. This problem can be overcome calculating the weight ω_i as the product of two terms, one linearly increasing with the pressure time and one decreasing with the dissimilarity between the current input I_* and the i-th example input I_i. Formally denote by $T_i[s]$, $1 \leq s \leq n$, $1 \leq i \leq E$ the pushing time of the s-th sensors in the i-th example and by P_i, O_i and V_i, $1 \leq i \leq E$ the joint angles of the robot in i-th example, the orientation (pitch roll and yaw) of the robot in the i-th example and the center of gravity velocity vectors for the i-th example respectively. Assume then an analogous definitions for the current input I_*. We defined the weight ω_i as

$$\omega_i = \begin{cases} 0 & if \ \bigvee_{s=1}^{n} (T_i[s] > 0) \wedge (T_*[s] = 0) \\ \alpha_i \cdot \beta_i & otherwise \end{cases}$$

$$\alpha_i = \prod_{s:T_i[s]>0} T_*[s] / T_i[s], \beta_i = \frac{1}{1+\sqrt{\gamma_i}}$$

$$\gamma_i = \sum_{s:T_i[s]=0} T^*[s]^2 + \|P_* - P_i\|^2 + \|O_* - O_i\|^2 + \|V_* - V_i\|^2$$

The structure of Eq. 2 emerges from practical experiments: several decreasing functions were tested and the one which appeared to give the most intuitive behavior was chosen. A deeper and more formal analysis will be conducted in future works. A more extended description of the algorithm will be provided in [1].

3. PROOF OF CONCEPT STUDY

A commercially available robot, Vstone's Vision4G (see Fig. 3), was employed to validate the feasibility of the approach. The context includes the velocity and orientation of the robot

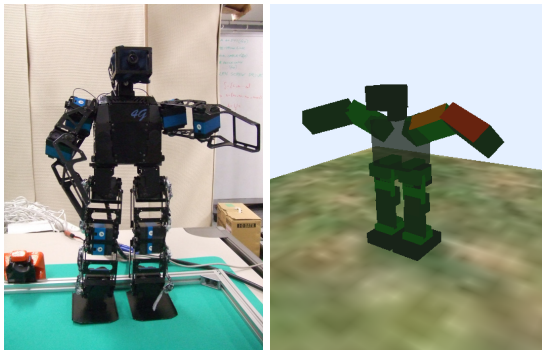

Figure 3: Photo of the robot employed in the experiments and its 3D representation.

and while these could be estimated using the internal gyroscopes and accelerometers, for simplicity we preferred to use a motion capture system (Eagle Digital System developed by Motion Analysis Corp). A stand up motion, a walking motion and a jump motion (developed with a rubberband pulling the robot from the top to ease the task) were successfully realized. In detail the jump motion was realized both with the touching approach and with a classical slider-based interface for comparison. Similar motions were obtained, respectively, in 17 minutes and in over 40 minutes. This preliminary result provides support to the hypothesis that motion development time can be reduced by introducing touch-based programming.

The data collected during the motion development were analyzed to see if the mapping from touch to change in joint angles could be formalized by a simple linear model obtained by linear regression. Denote by X the matrix which consists of the row vectors I_i, $1 \leq i \leq E$, and by Y the matrix having M_i, $1 \leq i \leq E$, as rows. Assuming a linear model $Y = XA + \epsilon$ where ϵ can be interpreted as Gaussian noise, we applied ridge regression, i.e. we calculated the matrix $A = (X^T X + \rho I)^{-1} X^T Y$ where ρ was tuned heuristically to 0.1 observing the cross validation error. Expressly we created three datasets, one of examples collected during the development of the jumping motion, one relative to the walking motions and one given by their union. Table 1 reports the average of the Euclidean distance of the predicted M_i (i.e. the value $I_i * A$) to the actual M_i obtained using just one of the data sets to construct the A matrix and testing the prediction on all the three data sets. For comparison the table reports the average distance (error) yielded when employing the k-Nearest Neighbor algorithm with the weighting schema we devised. We can clearly notice that on unseen data our algorithm outperforms linear regression since the latter tends to overfit the data.

User dependence was investigated asking six subjects (male computer science students, age in the range 23-27, mean 24.5, standard deviation 1.87) to develop the same motions, a walking and a kicking motion. Some tactile patterns seem to be naturally shared between the users. For instance in the default robot posture the arms are opened (see Fig. 3) and five out of six people pushed the arms from the back to close them. However the main finding was that different users gave different abstraction levels in providing touch

Table 1: Average errors in the output prediction by linear regression and by k-NN algorithm.

Training Dataset	Test Dataset	LR error	KNN error
JUMP	JUMP	0.2846	0.1985
JUMP	WALK	13.717	0.5938
WALK	JUMP	2.0567	1.0198
WALK	WALK	0.0314	0.0588
WALK	COMBINED	0.3626	0.2223
COMBINED	JUMP	0.3219	0.1425
COMBINED	WALK	3.7272	0.3006
COMBINED	COMBINED	1.5462	0.7778

instructions:

- a nearly fixed mapping from a small set of sensors to the joints; the context has little or no influence;
- a mapping on physical considerations (the joints are imagined to be "elastic"); in this case, the context, for instance the position of the ground, becomes crucial; this strategy strongly resembles the "pin and drag" model [8] used for computer animation, and the fact that this approach is taken intuitively by some users probably confirms the high usability of the pin and drag interface;
- a very high level representation of the motion, where for instance just the limb that should be moved is indicated by touching; at this level of abstraction a single touch corresponds to a motion primitive.

Quantitative analysis of the difference in the user data performed using multidimensional scaling will be provided in a work in progress that focuses on the development of motions for humanoid soccer robots [1] using touch.

4. CONCLUSIONS AND FUTURE WORK

In this paper we presented a system that utilizes tactile interaction to program robot motions. We believe touch to be a promising way of interacting with robots, even in the case of virtual robots. The focus of this work is not optimizing the motion performances (for instance far better walking can surely be obtained using specific concepts like the Zero Moment Point) but allowing inexperienced users to intuitively program robots, and, at the same time, gather insights in how humans employ touch to express their desired motion modifications. Figure 4 summarizes graphically the emergence of touch instructions. Users have an intended joint modification and, depending on some of the features of the physical context that they unconsciously perceive, they provide a tactile instruction. Our interface aims at constructing the inverse mapping from touch instruction and physical context to intended joint modification, by using examples consisting in tuples that include the touch instruction, the context and a directly communicated joint modification.

A proof-of-concept system was implemented and preliminary data analysis was performed on the collected data. As expected simple linear models failed to capture the structure of the mapping. Our algorithm instead revealed to perform well (at least in comparison to ridge regression) on unseen validation data and preliminary experiments suggest that

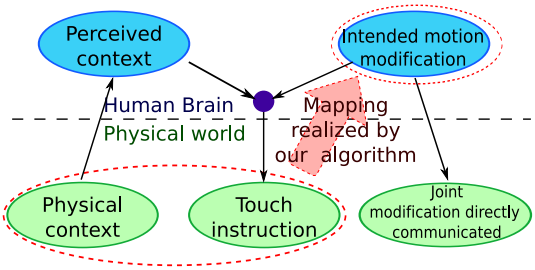

Figure 4: Conceptual schema of the generation of touch instructions.

the employed system can reduce the motion development time with respect to classical slider based interfaces. The presence of a strong user dependency of the touch protocol was identified using a limited number of users. Future works will need to employ more users to provide more statistically significant analysis. Furthermore the features of the physical context that influence the touch meaning must be analyzed more systematically and possibly extended with more dynamical features like the acceleration of the center of gravity or the location of the center of pressure.

5. REFERENCES

[1] F. DallaLibera, T. Minato, I. Fasel, H. Ishiguro, E. Pagello, and E. Menegatti. A new paradigm of humanoid robot motion programming based on touch interpretation. *Robotics and Autonomous Systems*, in press.

[2] G. Grunwald, G. Schreiber, A. Albu-Schffer, and G. Hirzinger. Touch: The direct type of human interaction with a redundand service robot. In *Proc. of the 10th IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN 2001)*, Bordeaux/Paris, France, 2001.

[3] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Trans. on Robotics*, 2008.

[4] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *2003 IEEE Intl. Conf. on Robotics and Automation (ICRA 2003)*, pages 3905–3910, Taipei, Taiwan, 2003.

[5] W. D. Stiehl, L. Lalla, and C. Breazeal. A somatic alphabet approach to sensitive skin for robots. In *2004 IEEE Intl. Conf. on Robotics and Automation (ICRA 2004)*, pages 2865–2870, New Orleans, USA, 2004.

[6] T. Takeda, Y. Hirata, and K. Kosuge. Hmm-based error recovery of dance step selection for dance partner robot. In *2007 IEEE Intl. Conf. on Robotics and Automation (ICRA 2007)*, pages 1768–1773, Roma, Italy, 2007.

[7] T. Wama, M. Higuchi, H. Sakamoto, and R. Nakatsu. Realization of tai-chi motion using a humanoid robot. In R. Jacquart, editor, *IFIP Congress Topical Sessions*, pages 59–64. Kluwer, 2004.

[8] K. Yamane and Y. Nakamura. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on visualization and computer graphics*, 9:352–360, 2003.