RoboJockey: Designing an Entertainment Experience with Robots

Shigeo Yoshida[*1], Takumi Shirokura[*2], Yuta Sugiura[*3], Daisuke Sakamoto[*1], Tetsuo Ono[*2], Masahiko Inami[*3], and Takeo Igarashi[*1]

[*1]The University of Tokyo
[*2]Hokkaido University
[*3]Keio University

Abstract—**An entertainment system, called "RoboJockey", for creating a robot's dance performances, was developed. RoboJockey consists of a multi-touch tabletop interface for multi-user collaboration, and it enables a user to choreograph a robot to dance by using a simple visual language. The visual language has two types of objects; robot and action. A robot object represents the real robot. RoboJockey supports a mobile robot and a humanoid robot. An action object represents the actual actions of robots. With RoboJockey, a user can coordinate the mobile robot's actions with a combination of back, forward, and rotating movements and coordinate the humanoid robot's actions with a combination of arm and leg movements. Every action is automatically performed to background music. RoboJockey was demonstrated at several places, and the users' behavior was observed. In this paper, we report the results of the observations and discuss the entertainment experience that RoboJockey gave to the users.**

Keywords—Robotic dance performance, entertainment experience with robots, multi-touch interface, multi-user collaboration, visual language.

I. Introduction

These days, robots can be seen entertaining people in environments such as amusement and theme parks, museums, shopping malls, and homes. Watching a robot dance is entertaining, but coordinating it is difficult. Moreover, it takes robotics engineers a long time to construct entertainment robots. Aiming to shorten that construction time, engineers have recently developed robot action-coordination interfaces. These interfaces, however, were developed for robotics experts who can construct robots on their own or have a strong background in robotics. Furthermore, it is even difficult for robotics experts to coordinate dances for robots. Naturally, that means robots are still difficult for end users (namely, non-experts) to use.

To address the above-described issues, in the present study, an interface—called "RoboJockey"—for easily controlling robots and providing new entertainment experiences to end users, who are not only creating a robot dance but also watching it, was developed (Figure 1). The interface contains a simple visual language that allows a user without any prior experience or knowledge in robotics to create a robot's actions and control a robot via the combination of those actions. The interface provides a feeling of being a "robot jockey," namely, a choreographer of a robot's dance, to the end user in a similar manner to that of a "disc jockey" or a "video jockey" selecting and playing music or videos for an audience. The key features of the RoboJockey interface are fourfold: first, easy coordination of robot motions by using a simple visual language; second, user-created actions are performed in real-time; third, robots automatically perform to music; and fourth, robots perform continuously and simultaneously.

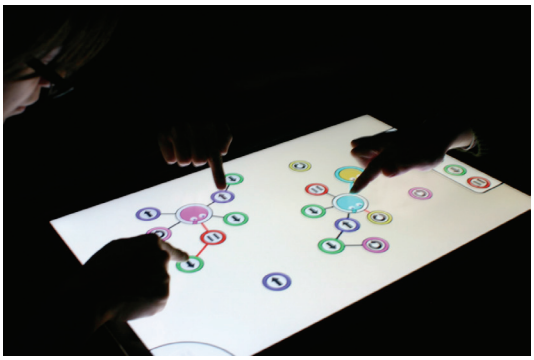

Fig. 1. RoboJockey interface; users coordinate robot performance by connecting action objects to robot objects.

II. Related Work

A. Programming interfaces for robots

The development of RoboJockey was inspired by the work reported in "Turning the Tables" by Taylor et al. [12]. They proposed an intuitive "visual jockey" interface on a tabletop interface. In the present study, their concept was extended to support physical robots, and a simple visual language for coordinating a robot's actions was created. Several visual languages and programming interfaces for robots are available. Lego Mindstorms NXT [6], which focuses on creating robot actions with a visual language, is one of the best-known interfaces. However, in this system, programming and execution are separate. For entertainment purposes, a real-time working system is required. Because the real-time readjustment prevents users from getting tired of creating robot motions and encourages users' motivations.

Topobo [8] is a robot that supports programming for specifying its motion, but it does not provide visual

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

representation of creating a robot's actions, making it difficult to modify or refine actions later. Hoffman et al. proposed a hybrid robot control system, which combines reactive expressive gestures and parametric behaviors, for live robotic stage performances [3]. This interface, however, is not for novice users and still has difficulties to control robot behaviors.

Coordinating dance actions for robots performing to music is even difficult for robotics experts. Studies on robot-controlling interfaces with multi-touch technologies have been done [10][11], but these interfaces are not designed for coordinating a robot's actions. They mainly focused on synchronous teleoperation of robots. In this work, we focused on creating a "robot dance" entertainment, and developing an interface which can easily coordinate with robot actions and music.

B. *Visual languages*

Max/MSP [7], which is a visual language designed mainly for audio and video processing, supports simultaneous editing and playback. Also, Hyperscore [2], a sketchpad like interface which supports novice user to compose music by visual language, was developed. However, they do not support physical objects that can move around an environment.

C. *Tabletop interfaces*

Tabletop systems are usually used for entertainment because they have multi-user collaboration capability. The reacTable [4] and Jam-O-drum [1] utilize tabletop systems to provide a novel musical experience. RoboJockey also focuses on providing such an experience to end users via a user-friendly robot-coordination interface, and it gives both experts and non-experts the chance to enjoy themselves by enabling them to coordinate a robot's actions while watching the robot dance.

III. Visual Language

Despite the growing population of robotics hobbyists, creating a robot's actions is still difficult. These hobbyists mainly focus on the entertainment aspect of robotics; however, it is still difficult for the end user to create a robots' actions. For mobile robots, a system has to detect objects in the real world because they move around the environment on wheels and have to avoid obstacles. For humanoid robots, the user has to manipulate many joints on the robots' bodies to simultaneously create actions. This manipulation is difficult and takes a long time even for robotics experts.

To solve these problems, RoboJockey uses a visual language, which is usually used for simplifying difficult tasks and many visual languages have been developed for practical computer-programming environments [5][9]. Creating music and visual arts is sometimes difficult for end users, and visual languages help them by hiding low-level operations, such as chord progression in music and creating image sources in visual arts. Visual languages are usually used in computer-science education. RoboJockey—with its simple visual programming language—simplifies the creation of a robot's actions for anyone.

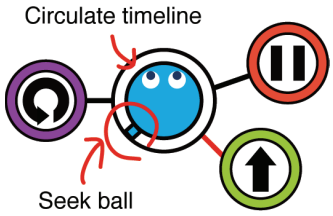

Fig. 2. Example of a visual program: the robot moves "forward" and continues until the seek ball crosses another action object, in this case, "Rotate-left".

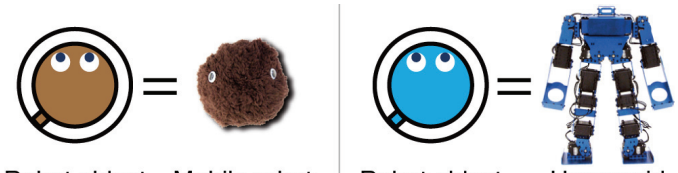

Fig. 3. Robot objects

Using RoboJockey, users coordinate robot actions by means of a simple visual language on a tabletop interface (Figure 1). The visual-language interface has two types of objects, namely, robot and action. The robot-action object has actual pre-designed motions. The end user does not have to concern him/herself with the actual creation of robotic actions. This robotic action will hide the most difficult tasks from the end user, who can coordinate a robot's actions by connecting the action object to the robot object. All objects are represented as icons that express the function of the object (Figure 2). The main contributions of the visual language are as follows:

- End users do not have to create low-level motions (which is the most difficult task in creating robot behaviors).
- Enabling end users to easily create complex behaviors of robots by the combination of predesigned motions.
- Automatic synchronization between music and a robot's dance motions.

A. *Robot*

The visual language is designed for two types of robots (Figure 3). One is a small mobile robot, which moves around its environment using two-degrees-of-freedom (DOFs) stepping motors; the other is a small humanoid robot (SMD, Tinywave), which has 16 DOFs. Although 16 DOFs seem not to be enough for human-like expression, it is still sufficient to create an entertaining robot dance. Robojockey interface treats these robots as a robot action, and these behaviors as an action object.

B. *Basic idea*

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

This article has been accepted for publication in IEEE Computer Graphics and Applications but has not yet been fully edited.
Some content may change prior to final publication.

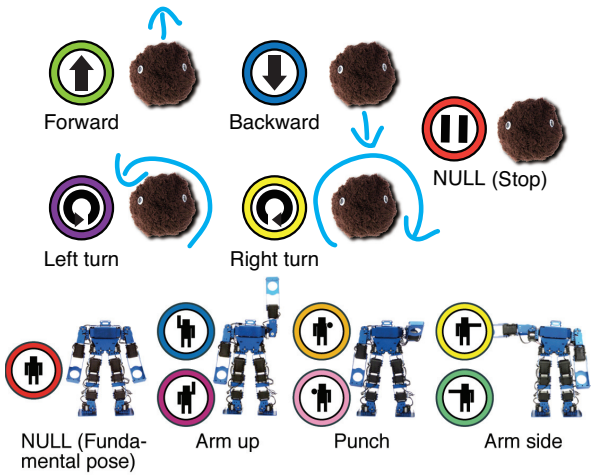

Fig. 4. A set of action objects for a mobile robot (top) and a humanoid robot (bottom)

A robot object has one "seek ball" circulating around it (Figure 2). The outer ring traced by the seek ball is called the "circulate timeline". When the seek ball crosses a line connected to an action object, the robot performs that action, which continues until the seek ball crosses another line. This mechanism enables the robot to continuously and simultaneously perform various actions with combinations discussed in the next sub-section. Connected objects can be disconnected from and reconnected to the robot object.

C. Primitive object

1) Robot object: A robot object represents a physical robot. The association between the robot object and the physical robot is shown by the color of the robot object (Figure 3). In this way, the user can easily see what robot is currently being controlled.

2) Action object: Action objects represent actual actions of the robot (Figure 4). A set of action objects is prepared for each robot in accordance with the robot's form. It is difficult (or sometimes impossible) to apply the same actions to a mobile robot and a humanoid robot. Each set of action objects has a NULL object, which shows the resting state of a robot. For mobile robots, the NULL object equals a stop action, and for humanoid robots, the NULL object equals a resting pose. The NULL object has a special meaning in the visual language, which is described in the following section.

D. Connection semantics

Multiple action objects can be connected to a robot object. Connections follow certain rules, which are categorized into three types, namely, serial, parallel, and synchronization.

1) Serial connection: All serially connected objects in one line are performed at the same time (Figure 5). In the case of the mobile robot, for example, the combination of "forward" and "rotation" objects results in a curved forward motion. The combination of two or three forward objects results in a two- or three-times faster forward motion. As for a NULL object, a NULL plus forward object will be a forward action, which means other object actions override the NULL object. In the case of a humanoid robot, for example, the combination of

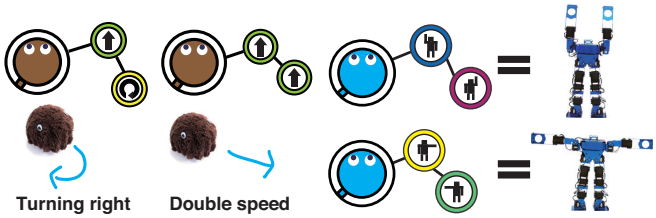

Fig. 5. Examples of serial connection

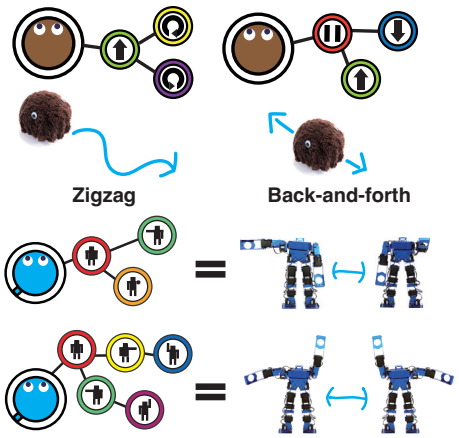

Fig. 6. Examples of parallel connection

"right arm up" and "left arm up" objects results in a "both arms up" motion, and that of "right arm side" and "left arm side" objects results in both arms stretched out to the side.

2) Parallel connection: Parallel connection results in actions occurring in turn. For example, the top left image in Figure 6 results in a mobile robot moving in a zigzag manner. This motion consists of forward curving left and right alternatively. As a result of these two motions occurring in turn, the robot will move in a zigzag manner. Back-and-forth motion uses one NULL object plus one forward and one backward object (Figure 6, top right). In this case, the program has two lines, one is a "NULL plus forward" action and the other is a "NULL plus backward" action. The NULL object plus another motion is useful for programming complex parallel connections.

This parallel-connection rule also applies to humanoid robots. Two example programs are shown in Figure 6. The timing of motion switching depends on the beat of the background music. The user can connect an action object by watching the robot move and the visual effect of the robot object repeatedly becoming bigger and smaller.

3) Synchronization of robots: Robot actions can be synchronized by connecting robot objects. Robot B's action synchronizing with robot A's action is shown in Figure 7. When a robot synchronizes with another robot, the color of the circulate timeline of the connecting robot object will change to the color of the host robot's face (e.g., the color of the timeline of robot B changed to the color of the face of robot A in Figure 7). When a user connects a robot object to another (e.g., robot B in Figure 7), the actions of the two connected robots will be synchronized. When the user detaches the robot object, the synchronization stops, and the robot that has no action objects will stop as well.

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

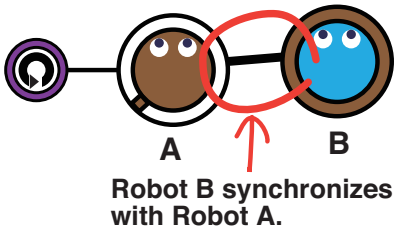

Fig. 7. Example of synchronization

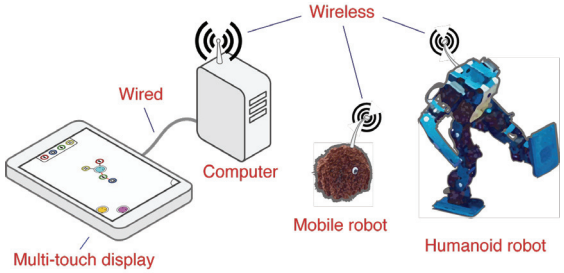

Fig. 8. System overview

E. Restrictions

Some connections are prohibited owing to hardware limitations of the robots and impossible semantic actions. In the case of humanoid robots, action objects for the same arm or the same actions cannot be connected on the same line. For both humanoid and mobile robots, the number of connections in one line is limited to three objects. Also, the connection with a mobile robot and a humanoid robot is prohibited because of the difference in their prepared action objects. Likewise, the visual language does not support explicit repeating actions (e.g., "loop") because the action is automatically repeated while the action object is connected.

IV. System Implementation

A detailed implementation of the RoboJockey is described as follows.

A. Hardware

RoboJockey consists of a multi-touch interface, robots, and a computer. The system produces music and beats and controls robots in rhythm to the music. An overview of the hardware configuration is shown in Figure 8.

1) Multi-touch interface: Multi-user collaboration is an important aspect of computer-entertainment systems. The multi-touch interface allows simultaneous control by multiple users. By allowing users to collaborate, the system provides an enhanced entertainment experience. RoboJockey uses a commercially available multi-touch interface and display, FlexScan T2351W by EIZO, for multi-user collaboration. The size of the interface is 23 inches.

2) Robots: It is necessary to create robot actions for each robot platform, a mobile robot and a humanoid robot. Because those robots have independent operating systems, the operation mechanism used on each robot platform must be considered. For mobile robots, it is easy to control and create robot actions because they can accept low-level commands from the computer. However, for humanoid robots, it is difficult to create actions in real time. All humanoid robot actions were thus designed in accordance with the number of action objects, including all combinations. Fifteen action patterns were designed for the humanoid robots, and they were all embedded in the robots.

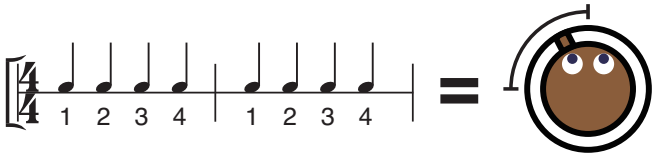

Fig. 9. Quarter circulate timeline includes eight beats, which means two bars in music.

B. Software

1) Visual-language interpreter: Users can create a robot's action with the visual language even when robots are performing; that is, the software has to accept the user-created program. To provide stable and natural interaction with users, the interpreter only reads an action object when the seek ball crosses it on the circulate timeline. The robot actions are cancelled by disconnecting action objects which are being executed. The interpreter produces commands (translated into a robot-readable style) for actual robot control with the user-created program.

2) Connection scheme: All action and robot objects can be connected to other objects. User can connect an object to another object from any direction, and it should be located outside of the object within a specified distance. A line will automatically appear between two objects if the connection is successfully established. Likewise, the user can cut off the action object by moving it outside the specified distance. All other operations, such as connecting an action object to another action object and connecting a robot object to another robot object, apply this mechanism for action synchronization.

3) Music and beat: One of the main features of RoboJockey is that robots automatically perform to music. The interface has a clock for beat and music. The beat is currently set as 32 bpm, which means that the seek ball moves and actions are executed and switched 32 times per minute. In other words, the timeline of the robot object is separated into eight bars, and one bar contains four beats (Figure 9). This mechanism is easy to understand for musicians and people who have basic knowledge of music. However, 32 bpm is a bit slow for most music and might sound strange to an audience. The music from the interface therefore matches the beat of music, but it will not be completely the same as the beat. The music may be two (64 bpm) or four (128 bpm) times faster than the system's beat. It will thus be natural music for the audiences.

A humanoid robot expresses a beat by moving its knees. A mobile robot does not have enough actuators for expressing rhythm; it just switches actions according to the interface's beat. Likewise, the interface represents the interface's beat. The robot object and currently running action objects repeatedly become bigger and smaller, in a similar manner to a heartbeat. For parallel-connected action objects, only the

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

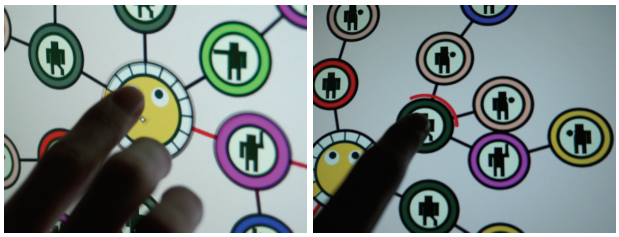

Fig. 10. Rotating a robot object with two fingers (left), Deleting child objects by holding down an object (right).

currently selected action will change its size, and each one will continuously keep changing its appearance in turn.

4) Auxilliary function for mobile robot: To create continuous robotic actions, auxiliary functions were developed for the mobile robot. When the robot is moving in an environment, it risks colliding with other mobile robots or straying off the stage. The interface detects these risk situations by using a web camera with visual markers. As a result, the robot can avoid these situations and give an uninterrupted performance.

5) Touch actions: RoboJockey aims to give the user a feeling of a "robot jockey," which is similar to a "disc jockey" or a "visual jockey," so the style of user-robot interaction is an important feature. RoboJockey employs a multi-touch display that allows the user to not only operate the interface intuitively and easily but also make an attractive performance as a "robot jockey." The touch actions supported by RoboJockey are summarized as follows.

a) Rotating objects in the manner of DJing: The user can rotate objects that are connected to a robot object, by rotating the robot object with two fingers. Rotating objects is useful for making a big visual program and pleasing in regard to performing a robotic dance visually. (Figure 10, left)

b) Tapping objects to adjust a connection: By double tapping an object, distances between its child objects can be adjusted. Making a big visual program is sometimes going to overlap objects and become overcrowded in a small space. This function arranges the connection at an appropriate distance and shows all objects that are connected to an object that the user tapped.

c) Long tapping for deleting child objects: By holding down an object for a few seconds, child objects that are connected to it can be erased. (Figure 10, right)

V. Demonstration and Observation

A. Mobile Robot

RoboJockey was demonstrated at a Japanese domestic HCI symposium as a pilot study. Audiences were not only HCI researchers and students but also non-HCI people. Especially, most audiences were non-robotics experts. More than 100 people experienced the interface, and more than 300 people watched its demonstration. We used three mobile robots in this demonstration.

Through a simple observation, many users seemed to enjoy the interface and robot performance. At the beginning of using the interface, participant users randomly coordinated robot actions. While they observed the robot actions, they learnt the visual language and how the robots move with the user-created actions. Within a few minutes, they could use the interface. Some of the users even created a complete dance performance to music.

The audience enjoyed watching the robots dance. Many people took photos and were impressed with the dance a novice user improvised for the robot. We did not confirm how the users understood the semantics and rules of the visual language. Even though they did not understand the entire system, they could coordinate robot actions and enjoyed the robot performances. The most important point regarding the RoboJockey interface (and most other entertainment systems) is whether users enjoy using the interface or not, even if they do not understand how it works. RoboJockey has the capability of accepting a user's vaguely defined operations and performing a robot dance continuously. In that way, it can provide a new entertainment experience to users.

Some of the users commented negatively on the presentation of the interface. For example, it was difficult to make out what action object was being executed in a parallel connection, and the multi-touch device failed to detect users' rapid movement. Because the RoboJockey interface is a prototype version, we will keep improving it in consideration of their comments. In particular, many points concerning the visual representation of RoboJockey need to be improved.

B. Humanoid Robot

RoboJockey was also presented at SIGGRAPH Asia 2012 Emerging Technologies. That time, three humanoid robots were used, and the user could synchronize the behavior of three robots by connecting the robot objects. During this exhibition, the user could operate the robots by using the touch actions described in IV.B.5. Furthermore, a "side step" action was added to the robotic actions. It represents a repetition of a step from one side to the other side. It allows the users to make a more dynamic robot dance. Moreover, we improved the performance of touch detections so as not to lose the user's rapid movement.

At first, the users did not notice the touch actions because no clues about the touch actions were installed. However, by teaching the users how to use them, the users emulated the instructions and started to use the touch actions actively. Although gestures are considered intuitive and easy to understand, visual information and effects should be added to the interface to notify the users about the touch actions.

Almost users could create complex robot performances by connecting action objects in serial and parallel way, and by synchronizing the behavior of robots. Moreover, some users attempted to place too many action objects onto robot objects by rotating the robot objects. There were three robot objects on the display and the area in which to place objects was limited. Other users rotated the robot object and turned it on its side where action objects are not connected, toward another user, and prompted him/her to connect new objects. In addition, alignment of action objects by double tapping was used when the users finished creating a robot dance and performed the

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

robot dance with object connections to the audiences. These touch actions improved the convenience of the collaboration work between users. Furthermore, in case of the user would like to change the behavior, they could erase part of the robot actions by holding their finger on an object where they did not like, and then they can choose one from list of action objects and connect it to where they like. This procedure enabled the user to more easily create a more spontaneous robot dance performance. All participants from all around the world enjoyed both creating and watching the user-created dynamic robot dances.

VI. Discussion

A. Overall

Overall, as the result of the observation, RoboJockey seemed to be able to give a new entertainment experience to the end users. How much the users understood the visual language was not confirmed by quantitative analysis. Even so, it was clear that the users really enjoyed playing with RoboJockey. In particular, this multi-touch interface entrained the collaborative operation of the users. Even though it lacks clues to notify users of possible touch actions, it helped collaborative operation and enabled users to create more spontaneous robot performances. Users all over the world, ranging from children to adults, enjoyed creating robot performances on their own. And the robot dance performances they created drew the attention of wider audiences. This demonstration of RoboJockey showed the possibility of new entertainment applications and experiences with robots.

B. Simplicity for a good entertainment system

RoboJockey employed visual language to create a robot's actions. This is first trial of the RoboJockey and many issues remain to be addressed. For example, the visual language does not support high-level functionality, such as programming loops and sensor feedbacks. These high-level functionalities are useful for making robots intelligent and giving them smart behaviors; however, we do not consider that these features are necessary for an entertainment system. That is to say, they will make the system and visual language difficult for the end-users to use. Because robots are just becoming part of peoples' daily lives, people who do not have knowledge of robots, who do not own them, or do not have any experience with them are still a majority. Even though we could make RoboJockey more complex and give it high-level features, RoboJockey seemed to be successfully accepted by the users in its present simple form. As a result, a wide range of users could understand how to manipulate the interface without needing detailed descriptions, and they could control not only mobile robots with a low degree of freedom but also humanoid robots with a high degree of freedom.

C. Limitations

Even though simplicity is important, RoboJockey has three key limitations in regard to future implementation. First, its visual language is too simple to create complex behaviors of robots, and it does not support extension or modification of actions. This might be a big limitation of the visual language, but it is an advantage of a robotic entertainment system. We did not aim to create practical robotic applications such as state-of-the-art humanoid robots working with people and rescuing people from disaster areas. We consider that demonstrating an entertainment application of robots has a significant meaning for the success of robotic technology; even though in Japan, which is a country well-known for production of robots, people rarely play with robots, and most Japanese people had never played with one. In other words, to understand what a robot can do for us and what a robot cannot do for us, it is necessary to play with them.

Second, robot-mounted sensors were not fitted to RoboJockey. These sensors are useful to make an autonomous robot, like Lego Mindstorms NXT. It is possible to incorporate them in RoboJockey, but the visual language must be significantly rewritten. The sensors may contribute to creating a more attractive entertainment system, but the visual language used in RoboJockey is not yet appropriate for such usage.

Third, the visual language does not support a "loop" function, because the robot object has a "circulate timeline," which makes the robot action continuous. A "micro-loop" mechanism, which enables the user to make a small-scale loop in the robot actions, might be useful for creating high-level actions. In future work, these issues will be considered while formal user studies and observations of end users are conducted.

VII. Conclusion

RoboJockey, an interface for creating entertaining robotic actions, was developed and demonstrated. This interface uses a visual language that is simple but powerful enough for creating complex robot actions. Two public demonstrations showed that experts and non-experts in robotics alike enjoyed using the interface to create dance routines for robots. We used not only the humanoid, but also the mobile robot to dance on a table. Though the type was different, these robots entertain not just the persons who are using the Robojockey system, but also the people around the table. By using RoboJockey, the users gained a new entertainment experience similar to being a "robot jockey." This result suggests that the RoboJockey system showed us a new way to interact with the robot as an entertainment system.

References

[1] Blaine, T. and Perkis, T. 2000. The Jam-O-Drum interactive music system: a study in interaction design. In Proceedings of the 3rd Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques (New York City, New York, United States, August 17 - 19, 2000). D. Boyarski and W. A. Kellogg, Eds. DIS '00. ACM, New York, NY, 165-173.

[2] Farbood, M.M., Pasztor, E., Jennings, K. 2004. Hyperscore: a graphical sketchpad for novice composers. Computer Graphics and Applications (January -February, 2004). IEEE, Vol.24. No.1. 50 - 54.

[3] Hoffman, G., Kubat, R., Breazeal, C. 2008. A hybrid control system for puppeteering a live robotic stage actor. in Proceedings of the International Symposium on Robot and Human Interactive Communication (Munich, Germany, August 01 - 03, 2008). RO-MAN '08. IEEE, 354-359.

[4] Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. 2007. The reacTable: exploring the synergy between live music performance and

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE

tabletop tangible interfaces. In Proceedings of the 1st international Conference on Tangible and Embedded interaction (Baton Rouge, Louisiana, February 15 - 17, 2007). TEI '07. ACM, New York, NY, 139-146.

[5] Ken Kahn. ToonTalkTM—An Animated Programming Environment for Children. Journal of Visual Languages & Computing, Volume 7, Issue 2, June 1996, Pages 197-217.

[6] Lego, Lego mindstorms NXT, http://mindstorms.lego.com/

[7] Max/MSP, http://cycling74.com/.

[8] Raffle, H. S., Parkes, A. J., and Ishii, H. 2004. Topobo: a constructive assembly system with kinetic memory. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 647-654.

[9] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. 2009. Scratch: programming for all. Commun. ACM 52, 11 (Nov. 2009), 60-67

[10] Rouanet, P., Oudeyer, P-Y., and Filliat, D. 2009. An integrated system for teaching new visually grounded words to a robot for non-expert users using a mobile device, in Proceedings of the IEEE-RAS International Conference on Humanoid Robots. Humanoids '09, IEEE, 391-398.

[11] Sugiura, Y., Fernando, C. L., Igarashi, T., Inami, M., Sugimoto, M., Withana, A. I., and Gota, K. 2009. An operating method for a bipedal walking robot for entertainment. In ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation (Yokohama, Japan, December 16 - 19, 2009). SIGGRAPH ASIA '09. ACM, New York, NY, 79.

[12] Taylor, S., Izadi, S., Kirk, D., Harper, R., and Garcia-Mendoza, A. 2009. Turning the tables: an interactive surface for vjing. In Proceedings of the 27th international Conference on Human Factors in Computing Systems (Boston, MA, USA, April 04 - 09, 2009). CHI '09. ACM, New York, NY, 1251-1254.

Shigeo Yoshida is a PhD student in the Graduate School of Interdisciplinary Information Studies at the University of Tokyo. Contact him at shigeodayo@cyber.t.u-tokyo.ac.jp.

Takumi Shirokura is a PhD student in the Graduate School of Information Science and Technology at Hokkaido University. Contact him at shirokura@complex.eng.hokudai.ac.jp.

Yuta Sugiura is a Project Assistant Professor in the Graduate School of Media Design at Keio University. Contact him at y-sugiura@kmd.keio.ac.jp.

Daisuke Sakamoto is a Project Lecturer in the Graduate School of Information Science and Technology at the University of Tokyo. Contact him at sakamoto@is.s.u-tokyo.ac.jp.

Tetsuo Ono is a Professor in the Graduate School of Information Science and Technology at Hokkaido University. Contact him at tono@ist.hokudai.ac.jp.

Masahiko Inami is a Professor in the Graduate School of Media Design at Keio University. Contact him at inami@kmd.keio.ac.jp.

Takeo Igarashi is a Professor in the Graduate School of Information Science and Technology at the University of Tokyo. Contact him at takeo@acm.org.

Digital Object Indentifier 10.1109/MCG.2015.1 0272-1716/$26.00 2015 IEEE