View-Based Teaching/Playback for Industrial Manipulators

Yusuke MAEDA and Yuki MORIYAMA

Abstract— **In this paper, we present a new method for robot programming: view-based teaching/playback. It was developed to achieve more robustness to changes of task conditions than conventional teaching/playback without losing its general versatility.**

Our method is composed of two parts: teaching phase and playback phase. In the teaching phase, a human operator moves a manipulator to achieve a manipulation task. All the movements of the manipulator are recorded. All the images of the teaching scenes are also recorded by a camera. Then, a mapping from the recorded images to the corresponding movements is obtained as an artificial neural network. In the playback phase, the motion of the manipulator is determined by the output of the neural network using camera images.

We applied this view-based teaching/playback to pushing tasks by an industrial manipulator. After multiple teaching demonstrations, pushing an object to the goal position was successfully achieved from some initial positions that were not identical to those in the demonstrations, without object models or camera calibrations.

I. Introduction

Robot programming is a very important part to use industrial manipulators. Conventional teaching/playback scheme is still widely used for its simplicity and generality; it only depends on their internal sensors such as rotary encoders and applicable to various tasks. Moreover, it is very reliable as far as task conditions, such as initial pose of the manipulated object, do not change. However, it is impossible to adapt to nontrivial variations in the initial pose of the object or unexpected fluctuations in the pose of the object during manipulation using only internal sensors.

Accordingly, detection of the object by visual sensing is becoming popular to adapt variations in the pose of the object. Model-based image processing such as feature extraction and pattern matching is performed to obtain the pose of the object accurately. In this method, however, we have to choose object-specific features to detect and localize the object. Hence it is cumbersome for operators to register the features for each object; that shows the limited general versatility of robot programming with model-based image processing. Moreover, operators have to calibrate camera parameters; it is another cumbersome task.

Now our motivation is to establish a new robot programming method so that we can achieve more robustness to changes of task conditions than conventional teaching/playback without losing its general versatility: "view-based robot programming" [1]. It uses PCA (Principal Component Analysis) [2] to perform image processing in view-based (or appearance-based) approach [3], which is not specific to the target object. It also uses the generalization ability of artificial neural networks to achieve robustness.

In [1], the method was successfully applied to pick-and-place and pushing operations by a robot hand with 8 degrees of freedom, but only in a virtual environment. Therefore in this paper, it is applied to operations by an actual industrial manipulator. Note that neither object models nor camera calibrations are necessary in our view-based method as well as conventional teaching/playback.

After overviewing related studies in the next section, we outline our proposed method for robot programming in Section III. In Section IV and V, each of the steps that constitute our method is described. In Section VI, our method is applied to pushing tasks by an industrial manipulator. Moreover, our method is extended to adapt to the changes of lighting conditions in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

Our approach can be categorized as programming by demonstration (PbD) [4]. Most of recent PbD studies in robotics deal with task-level programming, typically for humanoid robots, based on model-based image processing. On the other hand, our approach deals with motion-level programming for industrial manipulators based on view-based image processing.

Here we show some related works. Shibata and Iida dealt with reinforcement learning of box pushing by a mobile robot with an artificial neural network in a view-based approach [5]. Kobayashi et al. proposed a view-based method for reinforcement learning of box pushing by a manipulator with adaptive image resolution adjustment [6][7]. In contrast to these studies, we consider supervised learning in this paper.

View-based (or appearance-based) visual servoing (e.g., [8]) is also related to this work. This paper deals with robot programming and therefore differs from visual servoing studies in the following ways:

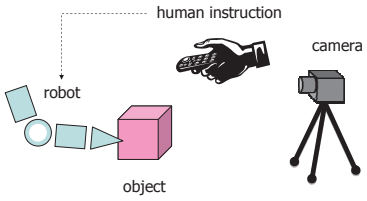
- How to teach the relationship between the image and the appropriate robot motion is explicitly described.
- The task is not just relative positioning to a target object but manipulation of the target object to a goal.

III. Outline of View-Based Teaching/Playback [1]

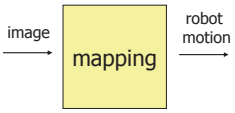
Our method is composed of two parts as well as conventional teaching/playback: teaching phase and playback phase.

Yusuke MAEDA is with Dept. of Systems Design, Div. of Systems Research, Faculty of Engineering, Yokohama National University. 79-5 Tokiwadai, Hodogaya-ku, Yokohama 240-8501 JAPAN maeda@ynu.ac.jp

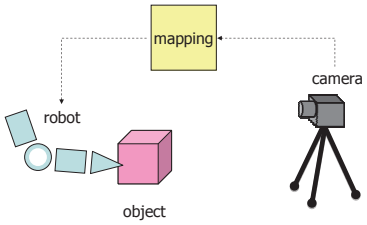
Yuki MORIYAMA is with Maeda Lab, Dept. of Mechanical Engineering, Div. of Systems Integration, Graduate School of Engineering, Yokohama National University.



(a) teaching by human operator



(b) obtaining mapping from image to robot motion



(c) autonomous playback with obtained mapping

Fig. 1. Outline of View-Based Teaching/Playback

In the teaching phase, a human operator moves a manipulator to perform a task (Fig. 1(a)). All the movements of the manipulator are recorded. All the images of the teaching scenes are also recorded by a camera.

Then, a mapping from the recorded images to the corresponding movements is obtained as an artificial neural network (Fig. 1(b)). The input of the neural network is a camera image, and its output is the desirable robot motion corresponding to the image.

In the playback phase, the motion of the manipulator is determined by the output of the neural network calculated from camera images (Fig. 1(c)). If the neural network is constructed appropriately, it can reproduce the motion of the manipulator in the teaching phase when the task condition is constant. Moreover, even when the condition is not constant (for example, the initial pose of the object varies), it may be able to drive the manipulator to complete the task thanks to the generalization ability of the neural network.

IV. Teaching Phase

A. Human Demonstration

In the teaching phase, a human operator moves a manipulator with an input device such as a teach pendant to perform a task. Here we call it *demonstration*.

One demonstration is enough to perform the task in a constant condition. However, our method has no advantage over conventional teaching/playback in such a constant condition. Hence we need two or more demonstrations in different conditions so that the generalization ability of the neural network can be utilized.

The pairs of the movement of the manipulator and the camera image are recorded throughout the demonstrations.

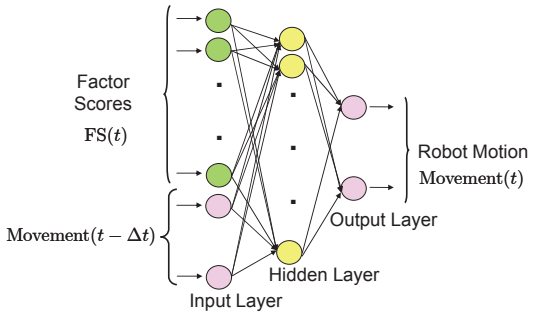


Fig. 2. Neural Network for Teaching/Playback

B. Image Compression by PCA

A camera image is composed of numerous pixel data and therefore it is not realistic to use raw pixel data as the input of the neural network. That is, image compression is necessary.

Here we use PCA (Principal Component Analysis) for all the recorded images as view-based image compression. Factor scores [2] for only a small number of principal components can reconstruct the original image approximately. Thus we use the factor scores as the input of the neural network instead of the raw pixel data.

We can use the method described in [9] to reduce the computation to obtain principal components.

C. Mapping by Neural Network

We use a feed-forward neural network as shown in Fig. 2 for mapping from image to robot motion. The neural network can be written as the following function:

$$\mathrm{Movement}(t) = f(\mathrm{FS}(t), \mathrm{Movement}(t - \Delta t)). \quad (1)$$

The output of the neural network is the movement of the manipulator at time t, $\mathrm{Movement}(t)$. The input of the neural network is factor scores of the current image for principal components, $\mathrm{FS}(t)$, and the movement of the manipulator in the previous time step, $\mathrm{Movement}(t - \Delta t)$, where Δt is the sampling time. The latter is added so that the neural network can recognize the "context" of the task; even if the camera image does not change, the neural network can change the motion of the manipulator. This is useful to switch robot motion (for example, from approaching to grasping) in some cases.

The weights of the neural network can be computed from image data and motion data in demonstrations by backpropagation with momentum (BPM) [10].

V. Playback Phase

In the playback phase, a robot motion is determined by computing the output of the neural network from the camera image (and the previous movement of the manipulator).

To finish the playback at the goal, we have to define a termination condition. Our method is view-based and therefore it is impossible to use the object pose explicitly to terminate the playback. Accordingly, we use the RMS error between factor scores of the current image and those

of the goal image in the termination condition. The error, ΔE_S, can be written as follows:

$$\Delta E_S = \sqrt{\frac{\sum_{i=1}^{N} |\mathrm{FS}(t)_i - \mathrm{FS}_{\mathrm{Goal}i}|^2}{N}}, \tag{2}$$

where $\mathrm{FS}(t)_i$ and $\mathrm{FS}_{\mathrm{Goal}i}$ are the factor scores of the current and goal image for the i-th principal component, respectively, and N is the number of principal components used in image compression. Note that the factor scores of the current image include implicit information on the current pose of the object.

Additionally we use the distance between the current and goal position of the hand of the manipulator, ΔE_P, which can be obtained from internal sensors of the manipulator, to terminate the playback.

Now the termination condition can be written as follows:

$$\Delta E_S < T_S \text{ and } \Delta E_P < T_P, \tag{3}$$

where T_S and T_P are threshold values. Equation (3) means that we terminate the playback when the camera image is almost identical to the image in the goal in the demonstration and the position of the hand becomes close to that in the goal.

VI. Application to Pushing Tasks

A. Experimental Setup

We prepared an experimental setup for teaching experiments as shown in Fig. 3.

We used a 6-axis industrial manipulator, RV-1A by Mitsubishi Electric. It accepts position commands from a controller PC via Ethernet. The controller PC has a Core 2 Quad Q9450 CPU (2.66GHz). We have another back-end PC with a Core 2 Quad 9650 CPU (3GHz) for offline computation including PCA and BPM.

A monochrome CCD camera, Flea2 FL2-03S2M by Point Grey Research, was installed for view-based teaching/playback. Note that we need neither intrinsic nor extrinsic calibration of this camera for our view-based teaching/playback. Another monochrome CCD camera, DMK21F04 by Imaging Source, was installed just for evaluation of playback results. Both of the cameras were connected to the controller PC via IEEE1394 interface.

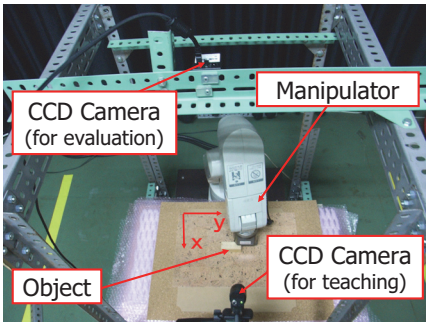


Fig. 3. Experimental Setup

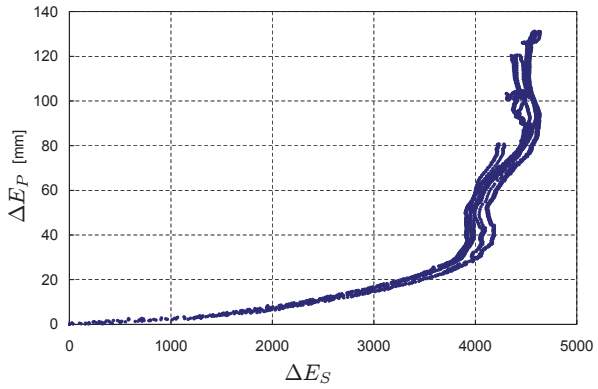
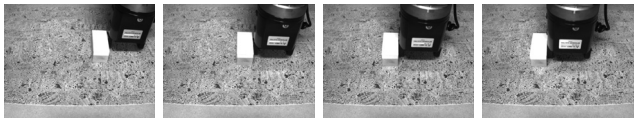
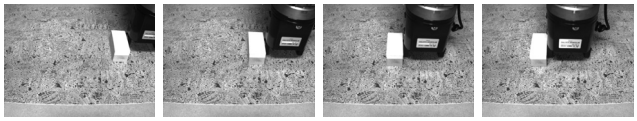


Fig. 4. Relationship between ΔE_S and ΔE_P in Some Cases



(a) demonstration 1



(b) demonstration 2

Fig. 5. Two Demonstrations in Teaching Phase

B. Pushing Tasks

We applied our view-based teaching/playback to quasi-static pushing [11]; the object was pushed by the hand of the manipulator to the goal. The object was a wood block whose size was $60 \times 30 \times 30$ [mm]. Note that we did not use the object information such as its dimension and shape in the experiments because our method is view-based.

In the teaching phase, a human operator moved the hand of the manipulator with a gamepad. The command from a thumbstick of the gamepad was mapped to 2D translation of the hand. Hence the movement of the hand can be expressed by a two-dimensional vector as follows:

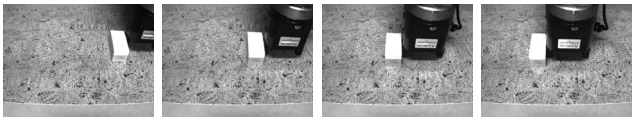
$$\mathrm{Movement}(t) = [\Delta\hat{x}(t), \Delta\hat{y}(t)]^T, \tag{4}$$

where $\Delta\hat{x}(t)$ and $\Delta\hat{y}(t)$ are displacements of the hand in X and Y directions, respectively. The pairs of the commanded movement and the camera image were recorded throughout demonstrations by the human operator. Grayscale images of 640×480 pixels were taken at 30 [Hz].

The PCA was performed on all the recorded images in the teaching phase and the first fifty principal components were calculated ($N = 50$).

The output of the neural network was two-dimensional (for $\mathrm{Movement}(t)$) and the input of that was 52-dimensional (50 for $\mathrm{FS}(t)$ and 2 for $\mathrm{Movement}(t - \Delta t)$). We used three-layered neural networks that have fifty neurons in their hidden layer. All the weights of the neural network were calculated by BPM with the momentum factor of 0.9. The

(a) from initial position of demonstration 1



(b) from initial position of demonstration 2

Fig. 6. Playback from Initial Positions of Demonstrations

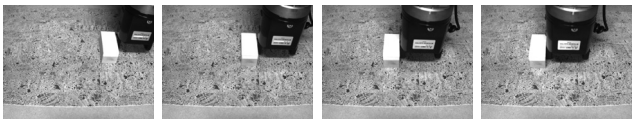


Fig. 7. Playback from Middle Point of Initial Positions of Demonstrations

BPM was iterated until the residual converged or the total computation time reached 300 [s].

In the playback phase, factor scores for the first fifty principal components were calculated for the current image. From the relationship between ΔE_S and ΔE_P in preliminary pushing experiments shown in Fig. 4, we set $T_S = 400$ and $T_P = 1.5$ [mm] for the termination condition (3). The control loop in the playback phase ran at 30 [Hz] , which was achieved by computing $\mathrm{FS}(t)$ using Intel SSE2 (Streaming SIMD Extensions 2) instructions in parallel threads.

C. Experimental Results

Fig. 5 shows two demonstrations in the teaching phase. The object was pushed from different initial positions (shown in the leftmost photos) to the same goal (shown in the rightmost photos).

Two playbacks from the initial positions in the demonstrations are shown in Fig. 6. The object was successfully carried to the goal position (shown in the rightmost photos). Positional errors of the object at the goal, i.e., the distances between the centroid of the object in the demonstration and that in the playback, were calculated from images taken by the camera for evaluation. The positional errors in the playback from the initial positions of demonstration 1 and 2 were 0.5 [mm] and 1.8 [mm], respectively. For comparison, we also performed conventional teaching/playback for demonstration 1 and 2, and the error was about 1.5 [mm] for both cases. These results show that our view-based teaching/playback can be used as a substitute for conventional teaching/playback. The error in conventional teaching/playback may be bigger than expected; one of the reasons for the error is our use of a slippery end-effector for pushing.

Fig. 7 shows playback from the middle point of the initial positions in the demonstrations. This situation was not taught in the teaching phase but the object was successfully carried to the goal position thanks to the generalization ability of the neural network.

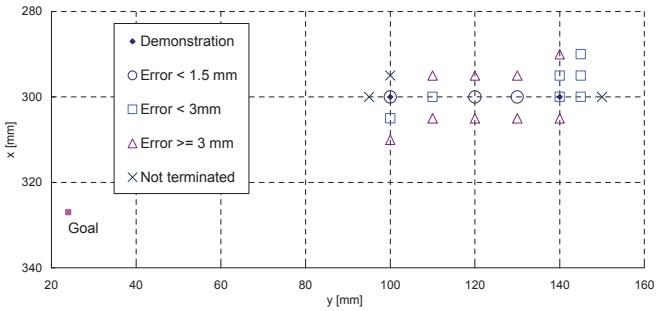


Fig. 8. Initial Positions and Results in Playback

Playback results from other initial positions can be found in Fig. 8. For simplicity, initial orientation of the object was constant in this experiment. The errors in the figure are positional errors of the object at the goal. The orientational errors of the object were negligibly small in twenty experiments and 2 [deg] in other two experiments.

In our method, when the initial position of the object in playback was inside those of the demonstrations, the object was carried to the goal successfully by interpolational motion of the manipulator generated by the neural network. The error was comparable with those of conventional teaching/playback in this case. That is, our view-based teaching/playback enabled the manipulator to adapt to variations in the initial positions of the object to some extent. However, extrapolational motion of the manipulator was not successful in carrying the object to the goal. To obtain more robustness to the fluctuations in initial positions and orientations of the object, we need more demonstrations.

The offline computation time in the teaching phase in this experiment was as follows:

- PCA: 163 [s]
- BPM: 73 [s]

for 817 images with 640×480 pixels obtained in the teaching phase.

D. Discussion

As shown in the previous subsection, our view-based teaching/playback enabled the manipulator to move the object to the goal even from some initial positions that were not identical to those in the demonstrations.

A reason for the success is that the demonstrated manipulations were similar; the hand pushed and slid the object to the goal in both of the demonstrations. The movement of the manipulator generated by the neural network will be interpolation or extrapolation of the demonstrations. If the manipulator must perform operations that are qualitatively different according to the task condition, we would need much more demonstrations for successful interpolation.

VII. Addressing Changes of Lighting Conditions

Our method is affected by changes of lighting conditions because it is view-based. However, robustness to the changes of lighting conditions is desirable in practical applications.

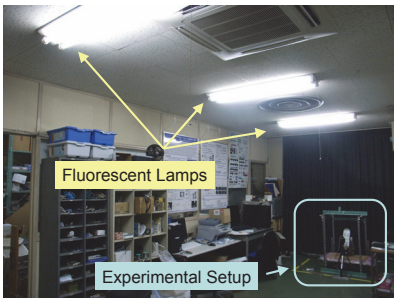
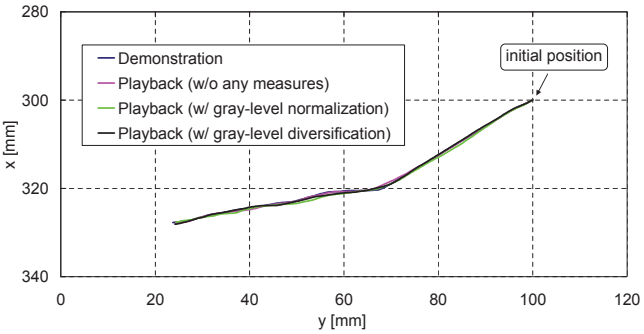


Fig. 9. Light Sources in Experiment Space

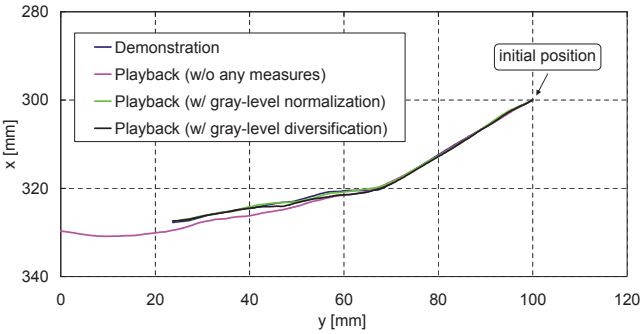
(a) original (b) darker

Fig. 10. Images in Different Lighting Conditions

Fig. 11. Playback Failure in Darker Condition



(a) in original lighting condition



(b) in darker lighting condition

Fig. 12. Paths of Manipulator Hand

Here we present the following approaches to address this problem:

- gray-level normalization in teaching and playback phases [1]
- gray-level diversification in teaching phase

We used two lighting conditions in the experiments: original and darker conditions. The teaching phase was carried out in the original lighting condition, where all of the six fluorescent lamps on the ceiling were turned on (Fig. 9). On the other hand, the playback phase was done both in the original and darker conditions. In the darker condition, only one fluorescent lamp on the ceiling was turned on. Examples of the image difference in these lighting conditions are shown in Fig. 10.

For evaluation of these approaches, we used demonstration 1 (Fig. 5(a)), which has 336 images, as teaching data. Without any measures against the change of lighting conditions, playback overran the goal and failed (Fig. 11).

A. Gray-level Normalization in Teaching and Playback Phases

We used gray-level normalization to address changes of lighting conditions in [1] for experiments in a virtual environment. Here we apply it to the actual robot system presented in Section VI-A. We adjust each pixel value of captured images by means of gamma correction as follows:

$$I_{\mathrm{norm}} = \left(\frac{I - I_{\mathrm{min}}}{I_{\mathrm{max}} - I_{\mathrm{min}}} \right)^{\gamma}, \tag{5}$$

where I is the gray level of the pixel ($[0, 1]$) and I_{norm} is the adjusted gray level. I_{min} and I_{max} are the minimum and maximum gray levels in the captured image, respectively. γ is determined so that $I_{\mathrm{norm}} = 0.5$ when I is the median of the gray levels of the captured image; that leads to gray-level normalization.

We perform the above gamma correction both in the teaching and playback phases so that the effect of changes of lighting conditions can be reduced.

B. Gray-level Diversification in Teaching Phase

Here we propose another approach to address changes of lighting conditions: gray-level diversification in the teaching phase. That is, we fabricate additional teaching images from a captured image by gamma correction as follows:

$$I_{\mathrm{div}} = I^{\gamma}, \tag{6}$$

where I is the gray-level of the pixel of the captured image and I_{div} is the changed gray-level. γ is set to various values in order to fabricate multiple teaching images with different gray-levels.

By using the captured teaching images and additional artificial images, we can obtain a neural network that is robust to changes of lighting conditions. We used four gamma values: $\gamma = 0.8,\ 0.9,\ 1.1$ and 1.2; that is, the number of teaching images was five times (1680) as many as captured images (336).

C. Experimental Results

Fig. 12 shows the paths of the hand in the teaching and playback. When the lighting condition was identical to that

TABLE I

ERRORS IN OBJECT POSE AT GOAL

	w/o any measures		w/ gray-level normalization		w/ gray-level diversification	
lighting condition	original	darker	original	darker	original	darker
positional error [mm]	1.0	(not terminated)	1.4	1.4	1.0	1.0
orientation al error [deg]	0	(not terminated)	0	0	0	0

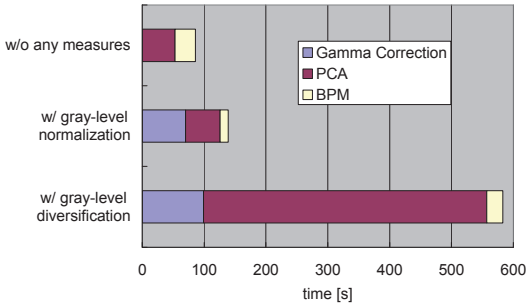


Fig. 13. Offline Computation Time

in the demonstration, the paths of the hand in the playback phase were almost identical to that in the teaching phase with or without any measures against the change of lighting conditions (Fig. 12(a)).

On the other hand, in the darker condition (Fig. 12(b)), the path in the playback without any measures went away from the demonstrated path and beyond the goal. However, the paths with gray-level normalization or diversification nearly followed the path in the teaching phase and ended at the goal. Thus both of the gray-level normalization and diversification can reduce the effect of the change of lighting conditions. Positional and orientational errors at the goal in this experiment are shown in Table I.

A potential problem in gray-level normalization is its on line computation cost. In our current implementation, the control loop in the playback ran at 30 [Hz] with gray-level normalization using a lookup table in parallel threads. However, there is little room for higher frame rate or image resolution. On the other hand, offline computational penalty is smaller than gray-level diversification (Fig. 13).

In gray-level diversification, additional computation is required only in the teaching phase and the control loop in the playback ran at 30 [Hz] with little difficulty. However, the offline computation cost became large especially in PCA (Fig. 13) because we had to process more teaching images.

VIII. CONCLUSION

In this paper, we demonstrated a new method for robot programming: view-based teaching/playback. It was developed to achieve more robustness to changes of task conditions, such as variations in initial positions of the object, than conventional teaching/playback without losing its general versatility. It requires neither object models nor camera calibrations for robot programming.

The view-based teaching/playback was applied to pushing tasks by an industrial manipulator. Using multiple demonstrations in the teaching phase, the manipulator moved an object successfully to the goal position in the playback phase, even from some initial positions different from those in the demonstrations. Two techniques for adaptation to changes of lighting conditions were also presented. They reduced the effect of changes of lighting conditions in the experiments.

There are many issues to be addressed in our future work. For example:

- *Evaluation of manipulation accuracy and its improvement.* Higher accuracy should be achieved for practical applications.
- *Application to various tasks.* We applied our method only to simple quasistatic pushing. It should also be applied especially to manipulation by grasping, which was tackled only in a virtual environment in [1].
- *Reduction of computation time.* In our current implementation, the control loop for the playback phase runs at 30 [Hz] for images of 640×480 pixels. If we need higher frame rate or image resolution, online computation time should be faster. GPU computing is a possible solution.
- *Emergency stop.* The system should be able to detect unexpected motions of the manipulator in the playback phase and stop it for fail-safe.

REFERENCES

[1] Y. Maeda, T. Nakamura, and T. Watanabe, "View-based teaching/playback for grasp and graspless manipulation," in *Proc. of Int. Conf. on Advanced Mechatronics*, 2010, pp. 75–80.

[2] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[3] Y. Matsumoto, M. Inaba, and H. Inoue, "View-based navigation using an omniview sequence in a corridor environment," *Machine Vision and Applications*, vol. 14, no. 2, pp. 121–128, 2003.

[4] A. Billard, S. Calinon, R. Dillman, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, ch. 59.2, pp. 1371–1394.

[5] K. Shibata and M. Iida, "Acquisition of box pushing by direct-vision-based reinforcement learning," in *Proc. of SICE Annual Conf.*, 2003, pp. 1378–1383.

[6] M. Kato, Y. Kobayashi, and S. Hosoe, "Optimizing resolution for feature extraction in robotic motion learning," in *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics*, 2005, pp. 1086–1091.

[7] Y. Kobayashi, M. Kato, and S. Hosoe, "Optimizing resolution for feature extraction in robotic motion learning," *J. of Robotics Society of Japan*, vol. 25, no. 5, pp. 770–778, 2007, (in Japanese).

[8] Q. Zhao, Z. Sun, F. Sun, and J. Zhu, "Appearance-based robot visual servo via a wavelet neural network," *Int. J. of Control, Automation and Systems*, vol. 6, no. 4, pp. 607–612, 2008.

[9] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[10] M. R. G. Meireles, P. E. M. Almeida, and M. G. Simões, "A comprehensive review for industrial applicability of artificial neural networks," *IEEE Trans. on Industrial Electronics*, vol. 50, no. 3, pp. 585–601, 2003.

[11] M. T. Mason, *Mechanics of Robotic Manipulation*. MIT Press, 2001.