Maeda and Nakamura *ROBOMECH Journal* (2015) 2:2
DOI 10.1186/s40648-014-0025-4

ROBOMECH Journal
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

View-based teaching/playback for robotic manipulation

Yusuke Maeda[1*] and Takahito Nakamura[2]

Abstract

In this paper, we study a new method for robot programming: view-based teaching/playback. The motivation of its development is to achieve more robustness against changes of task conditions than conventional teaching/playback without losing its general versatility. For proof of concept, the method was implemented and tested on a virtual environment.

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

We applied this view-based teaching/playback to pick-and-place and pushing by a robot hand with eight degrees of freedom in the virtual environment. Human demonstrated manipulation was successfully reproduced by the robot hand with our proposed method. Moreover, manipulation of the object from some initial positions that are not identical to those in the demonstrations was also successfully achieved with our method.

Keywords: Robot programming; View-based approach; Neural networks

Background

Conventional teaching/playback is still widely used in robot programming. It only depends on their internal sensors such as encoders for joint angles, and therefore it is simple and applicable to various tasks. Moreover, it is very reliable as far as task conditions, e.g. initial pose of the manipulated object, do not change. However, it is impossible for a robot to adapt to nontrivial variations in the initial pose of the object or unexpected fluctuations in the pose of the object during manipulation relying on only internal sensors.

Thus image-based detection of the object came into fairly widespread adoption to adapt variations in the pose of the object. Model-based image processing such as feature extraction and pattern matching is performed to localize the object accurately. In this method, however, how to detect an object is specific to the target object. Therefore it is cumbersome for operators to register object models with the detection system and the general versatility of this method is limited. Moreover, model-based image processing usually requires camera calibration and appropriate lighting.

Now we are motivated to develop a new robot programming method for achievement of more robustness against changes of task conditions than conventional teaching/playback without losing its general versatility. We study "view-based robot programming" in this paper. It uses PCA (Principal Component Analysis) to perform image processing in view-based (or appearance-based) approach [1], which is not specific to the target object. It also uses the generalization ability of artificial neural networks to obtain robustness. The method is applied to robotic manipulation in a virtual environment for proof of its concept.

Here we show some related works. Study on vision-based robot programming from human demonstrations has a long history (e.g., [2-4]). Recent efforts on programming by demonstration can be found at [5,6]. However,

*Correspondence: maeda@ynu.ac.jp
[1]Faculty of Engineering, Yokohama National University, 79-5 Tokiwadai, Hodogaya-ku, 240-8501 Yokohama, Japan
Full list of author information is available at the end of the article

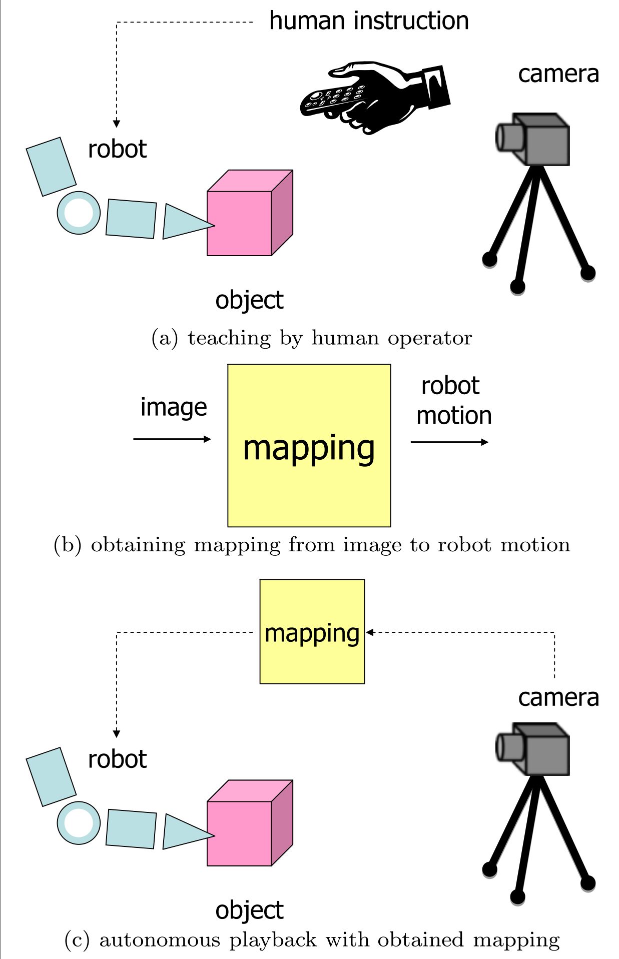

human instruction
camera
robot
object
(a) teaching by human operator
image
mapping
robot motion
(b) obtaining mapping from image to robot motion
mapping
camera
robot
object
(c) autonomous playback with obtained mapping

Figure 1 Outline of view-based teaching/playback.

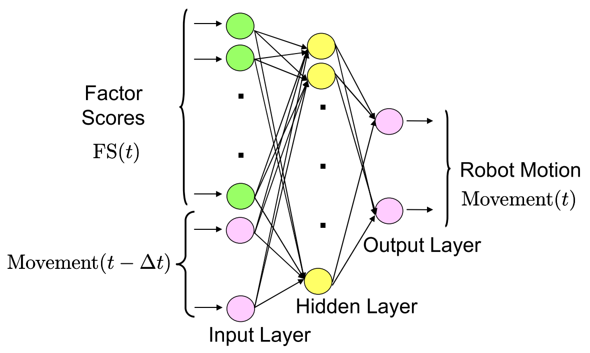


Figure 2 Neural network for teaching/playback.

most of them use not view-based, but target-specific model-based techniques for image processing.

The view-based (or appearance-based) image processing was applied to mobile robot navigation (e.g., [1]) and visual servoing of robot manipulators (e.g., [7]) in many studies. In these applications, the objective is relative positioning of a robot with regard to a target. On the other hand, robotic manipulation requires positioning of not only a robot but also a manipulated object.

There are few studies on view-based robotic manipulation. Zhang et al. presented a method for positioning of robots with eye-in-hand cameras [8] for grasping objects. In their study, assuming that coarse positioning of a robot hand was completed, fine positioning of the hand with regard to a target object was achieved by view-based neuro-fuzzy control. Their method is for relative positioning of only a robot with regard to a target, too.

Shibata and Iida dealt with reinforcement learning of box pushing by a mobile robot with view-based image processing [9]. They focused on learning from scratch and therefore the box pushing task was rather simple; not to carry the box to a goal, but just to push the box. Kobayashi et al. also proposed a view-based method for reinforcement learning of robotic manipulation: pushing an object to a goal by a manipulator [10,11]. They focused on learning from scratch, too, and therefore the robot motion is rather coarse: movement in the four cardinal directions in 50 [mm] steps. In contrast to these studies on view-based reinforcement learning, we consider view-based supervised learning of robotic manipulation in this paper.

In the next section, we outline our proposed method for robot programming. In Section 'Teaching phase' and 'Playback phase', each of the steps that constitute our method is described in a general form. In Section 'Results', our method is applied to pick-and-place, a typical grasp manipulation, and pushing [12], a typical graspless (or nonprehensile) manipulation, in a virtual environment. The concrete details of our method for this implementation are also described. Some discussions on our proposed method are made in Section 'Discussion'. Finally, we conclude this paper in Section 'Conclusion'.

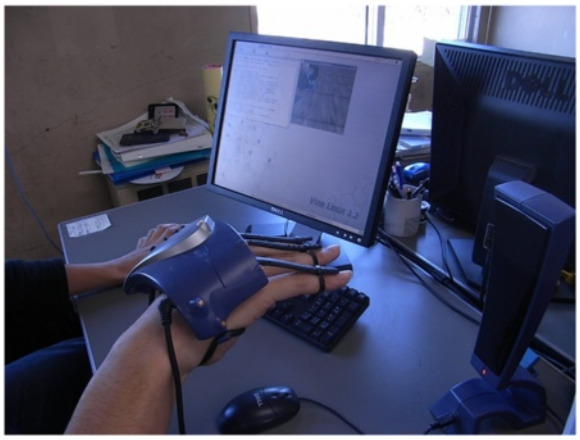


Figure 3 Experimental setup.

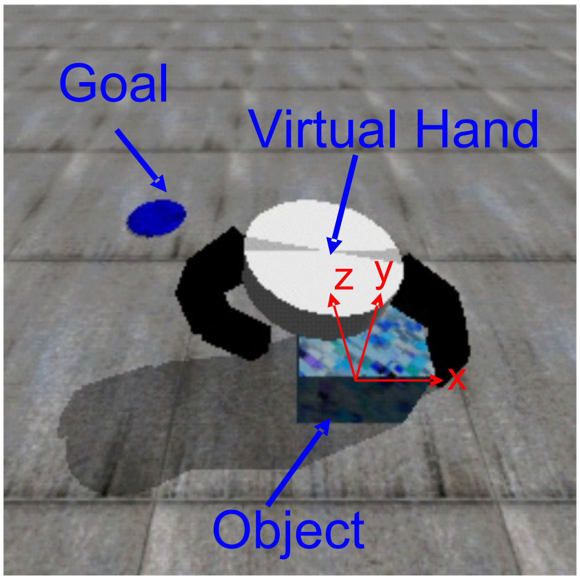


Figure 4 Virtual environment.

Methods

Outline of view-based teaching/playback

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

In the teaching phase, a human operator commands a robot to perform a manipulation task (Figure 1a). All the movements of the robot are recorded. All the images of the teaching scenes are also recorded by a camera.

Then, a mapping from the recorded images to the movements is obtained as an artificial neural network (Figure 1b). The input of the neural network is a scene image, and its output is the desirable robot motion corresponding to the image. The details of the teaching phase are described in generalities in Section 'Teaching phase'.

In the playback phase, the motion of the robot is determined by the output of the neural network calculated from scene images (Figure 1c). If the neural network is constructed properly, it must be able to reproduce the original robot motion demonstrated in the teaching phase as far as the task condition is unchanged. Moreover, even if the condition is changed (for example, the initial pose of the object varies), the neural network may be able to drive the robot to complete the task due to its generalization ability. The details of the playback phase are described in generalities in Section 'Playback phase'.

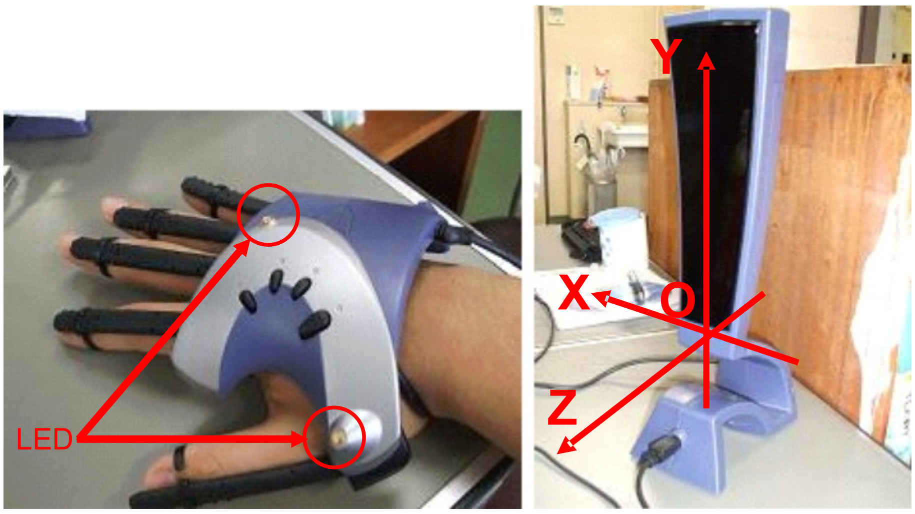
Teaching phase

Human demonstration

In the teaching phase, a human operator commands a robot with an input device such as a teach pendant and a data glove to perform a manipulation task. Here we call it *demonstration*.

One demonstration is sufficient to reproduce the task as far as the task condition is constant. However, our method has no advantage over conventional teaching/playback in such a constant condition. Thus we should perform two or more demonstrations in different conditions to adapt to changes of task conditions with the help of the generalization ability of neural networks.

The pairs of the movement of the robot and the scene image taken by a camera were recorded throughout the demonstrations. The scene images were grayscaled for compaction.



(a) P5 Glove (b) Receptor Tower

Figure 5 Data glove.

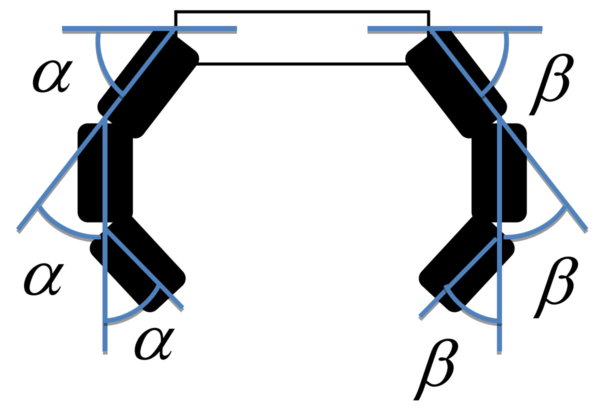


Figure 6 Finger bending.

Denoising and Gray-Level Normalization

Real scene images captured by a camera often have a noise. Here we use a 3 × 3 median filter to reduce the influence of image noise such as salt-and-pepper noise.

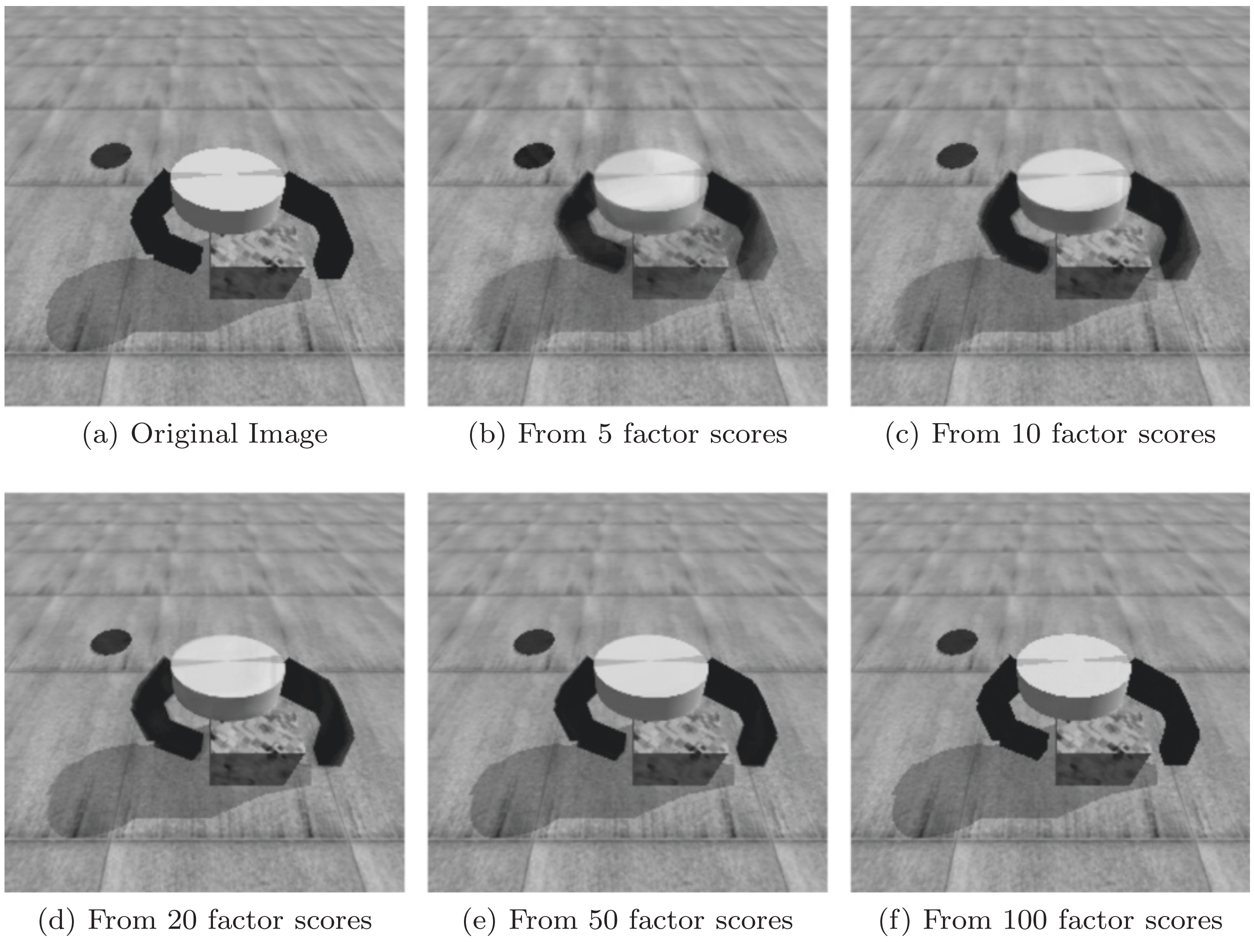
The real scene images are also affected by the changes of lighting conditions. Here we use gray-level normalization to reduce the influence of the lighting conditions. Each pixel value of grayscaled scene images is adjusted by means of gamma correction [13] as follows:

$$I_{\text{norm}} = \left(\frac{I - I_{\min}}{I_{\max} - I_{\min}} \right)^{\gamma}, \tag{1}$$

where I is the original gray level of the pixel ($I \in [0,1]$) and I_{norm} is the adjusted gray level. $I_{\min}$ and $I_{\max}$ is the minimum and maximum gray level in the original image, respectively. γ is determined so that $I_{\text{norm}} = 0.5$ when I is the median of the gray levels of the original image. Note that the value of γ is determined for each of scene images.

Image compression by PCA

A scene image vector $\boldsymbol{I}$ after median filtering and gray-level normalization is composed of numerous pixel data and therefore it is not realistic to use the raw pixel data as the input of the neural network. Here we use PCA (Principal Component Analysis) for all the scene images as



(a) Original Image
(b) From 5 factor scores
(c) From 10 factor scores
(d) From 20 factor scores
(e) From 50 factor scores
(f) From 100 factor scores

Figure 7 An Example of reconstructed images from factor scores.

view-based image compression as used in [8]. Even a small number of factor scores [14] can reconstruct the original image approximately as follows:

$$\boldsymbol{I} = \boldsymbol{I}_{\text{avg}} + \sum_{i=1}^{N_{\text{pixel}}} \text{FS}_i \boldsymbol{u}_i \approx \boldsymbol{I}_{\text{avg}} + \sum_{i=1}^{N_{\text{PC}}} \text{FS}_i \boldsymbol{u}_i, \qquad (2)$$

where $\boldsymbol{I}_{\text{avg}}$ is the average of scene image vectors in the demonstrations, N_{pixel} is the number of pixels of a scene image, FS_i are factor scores, $\boldsymbol{u}_i$ are principal components, and N_{PC} is the number of principal components to be used for approximation ($N_{\text{PC}} \ll N_{\text{pixel}}$). Thus we use the factor scores, $\text{FS}_1, \ldots, \text{FS}_{N_{\text{PC}}}$, as the input of the neural network instead of the raw pixel data.

We can use the technique described in [15] to reduce the computation to obtain principal components $\boldsymbol{u}_i$. It is applicable when N_{PC} is less than or equal to the number of demonstration images.

Mapping by neural network

We use a three-layered feedforward neural network as shown in Figure 2 for mapping from image to robot motion. The activation function for the units in the neural network is the standard sigmoid function. The neural network can be written formally as the following function:

$$\text{Movement}(t) = f(\text{FS}(t), \text{Movement}(t - \Delta t)). \qquad (3)$$

The output of the neural network is the movement of the robot at time t, Movement(t). The input of the neural network is factor scores of the current scene image, FS(t), and the movement of the robot in the previous time step, 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 scene image does not change, the neural network can change the motion of the robot. This is useful to switch robot motion (for example, from approaching to grasping) in some cases.

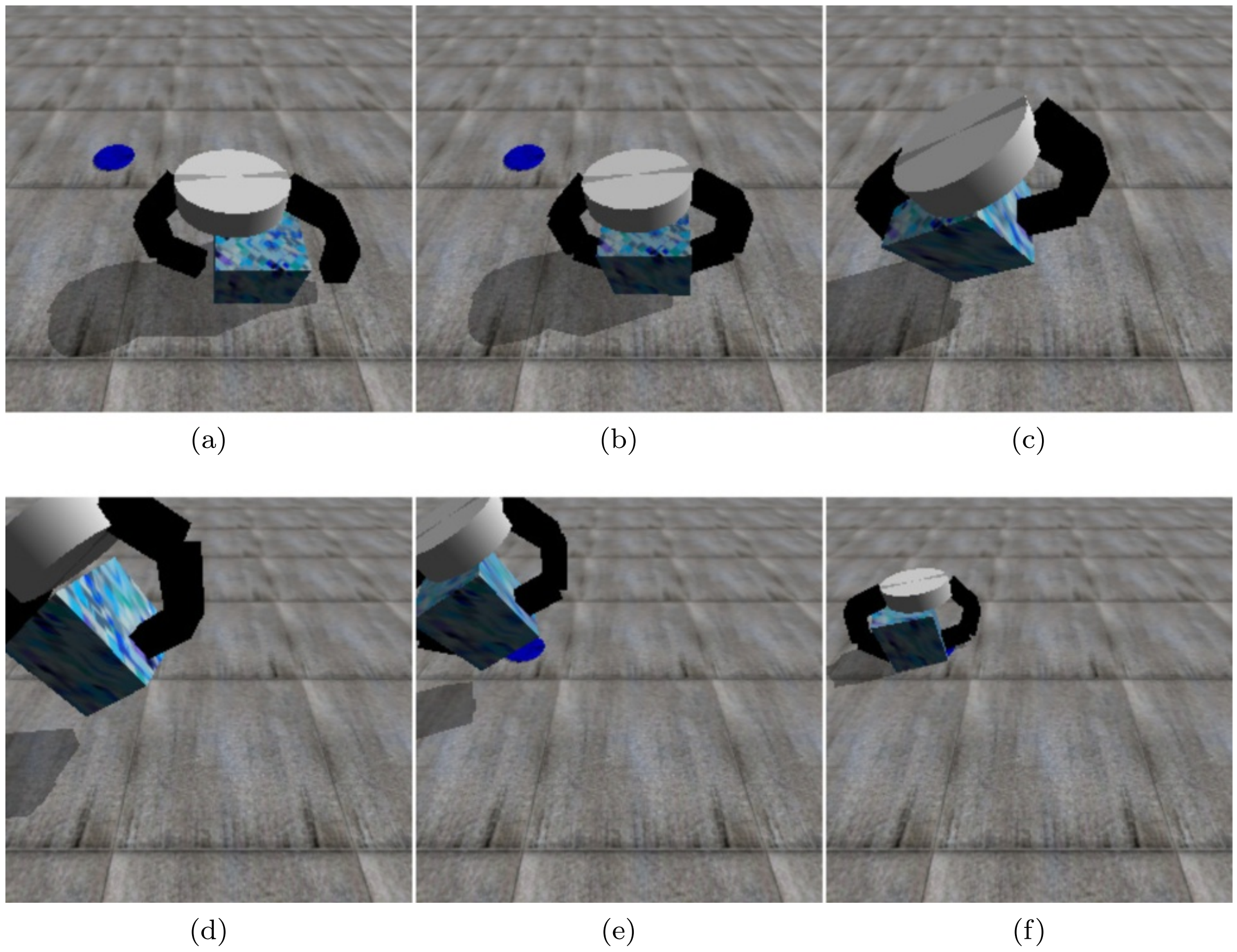


Figure 8 Demonstration of pick-and-place.

The weights of the neural network can be computed from image data and motion data in demonstrations by backpropagation with momentum (BPM) [16].

Playback phase

In the playback phase, a grayscaled scene image is obtained and denoised by 3×3 median filter as in the teaching phase. Then the gray-level of the image is normalized using Equation (1) as in the teaching phase, too. Factor scores FS_i for the normalized scene image $\boldsymbol{I}$ can be computed as follows:

$$\mathrm{FS}_i = \boldsymbol{u}_i^T \left(\boldsymbol{I} - \boldsymbol{I}_{\mathrm{avg}}\right). \quad (4)$$

Then the robot motion is determined by computing the output of the neural network from the factor scores (and the previous movement of the robot) using Equation (3).

The above procedure is repeated until the manipulation task is completed.

Results

Virtual environment

We prepared a virtual environment as shown in Figure 3 for proof of the concept of our view-based teaching/playback. A virtual robot hand (Figure 4) was created on a dynamics simulator, ODE (Open Dynamics Engine) [17]. The virtual hand was driven by a human operator with a data glove (Essential Reality P5 Glove, Figure 5). A Linux PC with a Pentium 4 running at 3.2 [GHz] was used for the virtual environment.

The position and orientation of the virtual hand were PD-controlled using those of the data glove as reference. The hand had two fingers: a “thumb” and an “index finger” (Figure 6). Each of the fingers had three joints, which were P-controlled so that the three joint angles were equal, using the bending sensor value of the corresponding glove finger as reference. Thus the virtual hand has eight degrees of freedom in total (3 DOF for position, 3 DOF for orientation and 2 DOF for finger bending).

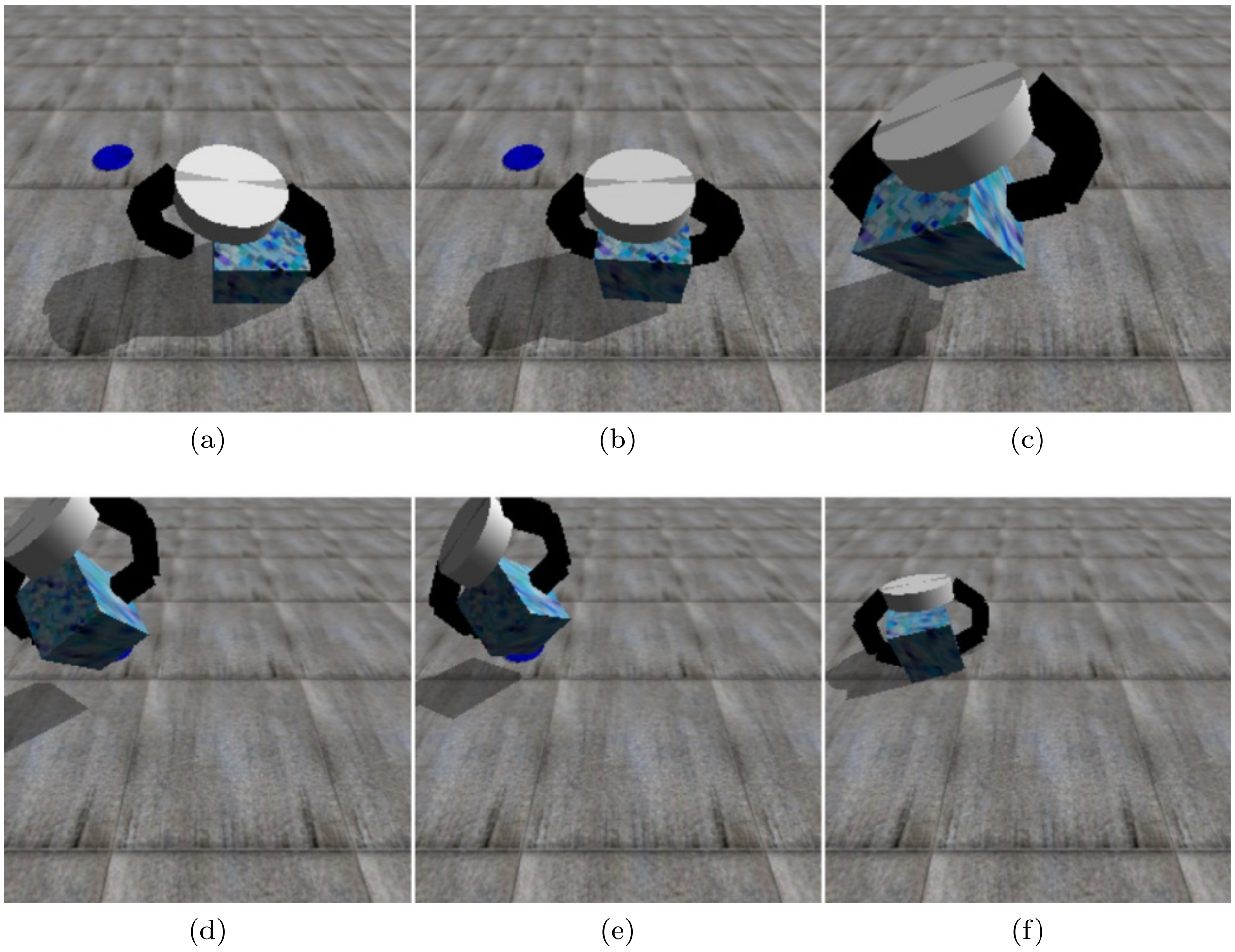


Figure 9 Playback of pick-and-place.

In the virtual environment, the human operator can drive the virtual robot hand with the data glove and perform dynamic manipulation of objects. Scenes of the virtual environment can be obtained as color images of 256×256 pixels by window capture instead of using a camera.

Pick-and-place and pushing

We applied our view-based teaching/playback to pick-and-place and pushing. The virtual object was a cube with edges of length 1 as shown in Figure 4. The goal position of the object is also shown in Figure 4 as a circle of 0.15 radius. The center of the goal circle is distant -0.9 in X-direction and 0.9 in Y-direction from the initial position of the center of the object. Note that we do not have to know the object information such as its dimension and shape because our method is view-based.

In the teaching phase, a human operator moved the virtual hand with the data glove. The movement of the hand can be written as an eight-dimensional $[0,1]$-normalized vector as follows:

$$\text{Movement}(t) = \left[\Delta\hat{\boldsymbol{x}}(t)^T, \Delta\hat{\boldsymbol{\theta}}(t)^T, \hat{\boldsymbol{b}}(t)^T\right]^T, \qquad (5)$$

where

$$\begin{aligned}\Delta\hat{\boldsymbol{x}}(t) &= \frac{\boldsymbol{x}(t) - \boldsymbol{x}(t-\Delta t)}{2\Delta x_{\max}} + \frac{1}{2} \\ \Delta\hat{\boldsymbol{\theta}}(t) &= \frac{\Delta\theta(t)\hat{\boldsymbol{K}}(t)}{2\Delta\theta_{\max}} + \frac{1}{2} \\ \hat{\boldsymbol{b}}(t) &= \frac{\boldsymbol{b}(t)}{b_{\max}}.\end{aligned}$$

$\boldsymbol{x}(t)$ is the position of the hand at time t; $\Delta x_{\max}$ is the upper limit of the change of $\boldsymbol{x}(t)$ in Δt; $\hat{\boldsymbol{K}}(t)$ is the equivalent axis of a finite rotation [18] of the hand between time $t - \Delta t$ and t; $\Delta\theta(t)$ is the amount of the rotation; $\Delta\theta_{\max}$ is the upper limit of the rotation in Δt; $\boldsymbol{b}(t)$ is the angles

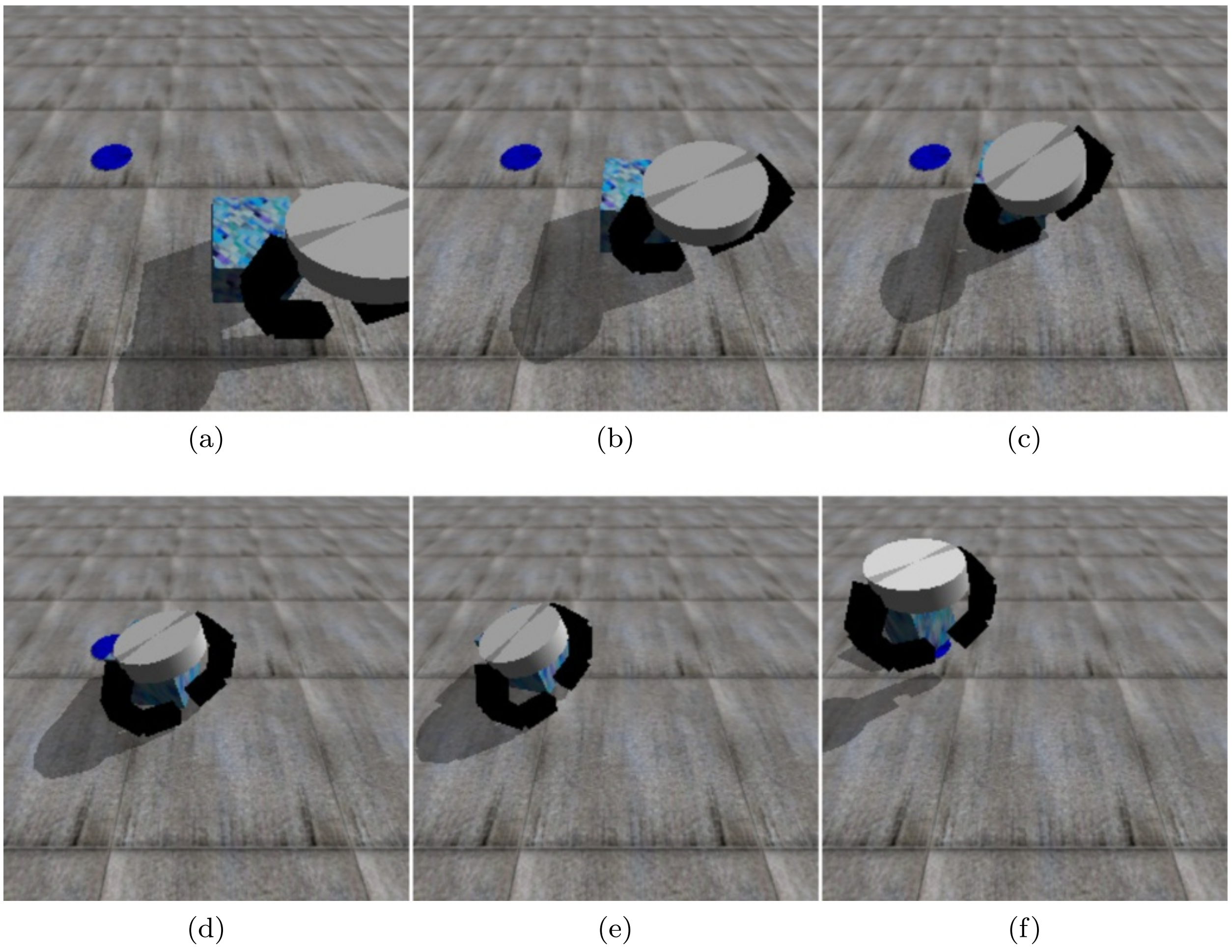


Figure 10 Demonstration of pushing.

of the flexion of the thumb and the index finger; $b_{\max}$ is the upper limit of the flexion angle of the fingers. The pairs of the movement and the scene image were recorded throughout demonstrations by the human operator.

In pick-and-place, the operator pinched the object by the two fingers of the virtual hand, lifted it up, and carried it to the goal position. In pushing, the operator pushed the object by the tips of the two fingers of the virtual hand and carried it to the goal position.

The PCA was performed on all the scene images in the teaching phase. In this case, one hundred factor scores can reconstruct an original image well (Figure 7). Thus the first one hundred principal components were calculated ($N_{\mathrm{PC}} = 100$).

The output of the neural network was eight-dimensional (for Movement(t)) and the input of that was 108-dimensional (100 for FS(t) and 8 for Movement($t - \Delta t$)). We used three-layered neural networks that have one hundred neurons in their hidden layer. All the weights of the neural networks were determined by BPM.

In the playback phase, one hundred factor scores were calculated for each scene image. The playback manipulation was terminated when the object touched the goal circle.

Simple playback of demonstration

Figure 8 shows a demonstration of pick-and-place in the teaching phase. The object was carried to the goal position and one hundred scene images were taken in the demonstration. Figure 9 shows a playback of the demonstration in our proposed method. The object was successfully picked up and placed at the goal position.

Figure 10 shows a demonstration of pushing in the teaching phase. The object was carried to the goal position and one hundred scene images were taken in the demonstration. Figure 11 shows a playback of the demonstration in our proposed method. The object was successfully pushed to the goal position.

These results show that our view-based teaching/playback can be used as a substitute for conventional

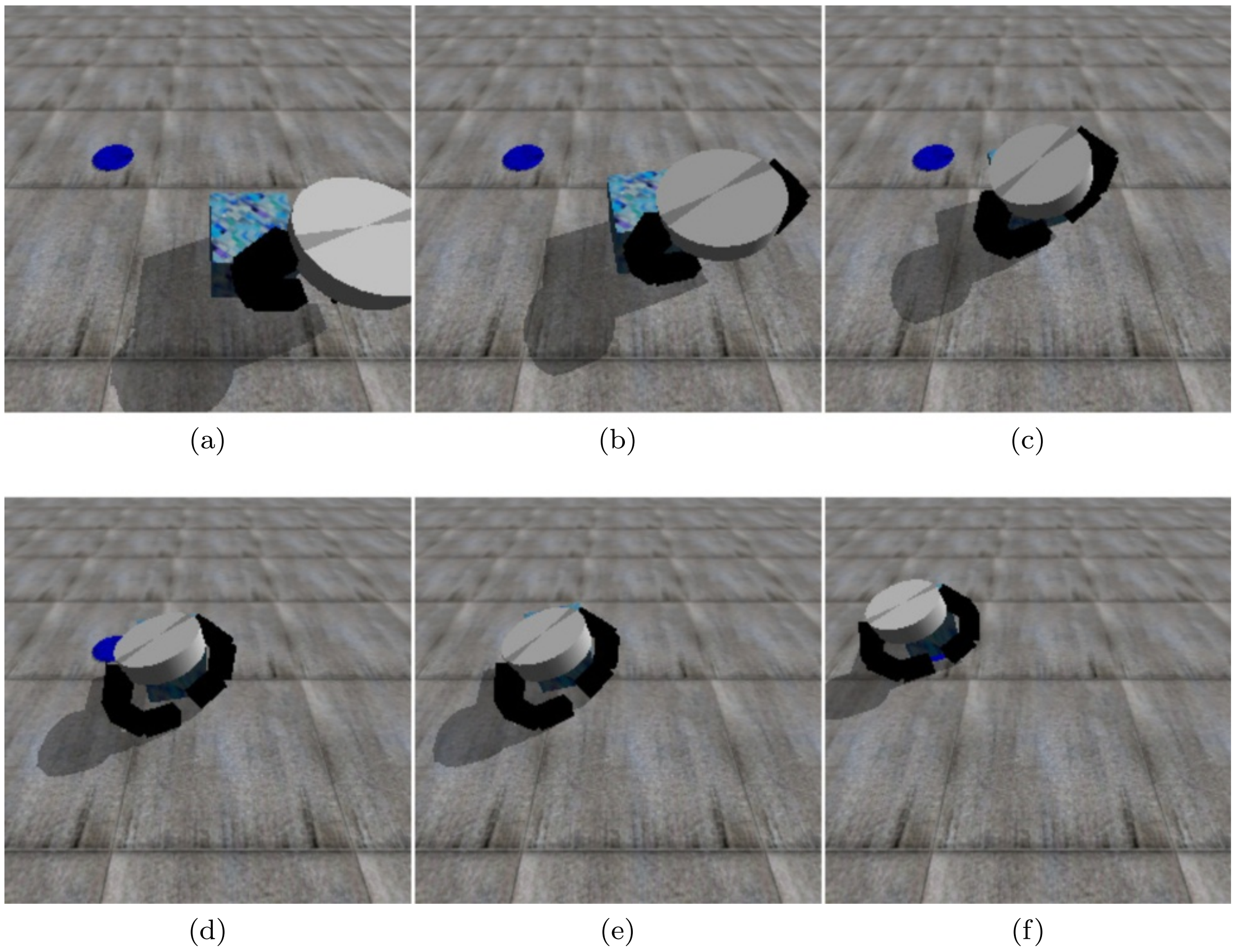


Figure 11 Playback of pushing.

Table 1 Offline computation time (100 images in total)

Process	Time [min]
Median Filtering & Gray-Level Normalization	0.5
PCA	4
BPM	10

Table 2 Offline computation time (900 images in total)

Process	Time [min]
Median Filtering & Gray-Level Normalization	2
PCA	23
BPM	142

teaching/playback, even with partial visual occlusion of the object by the hand. Offline computation time for these experiments is shown in Table 1.

The cumulative contribution ratio of the first one hundred principal components was 100% for the scene images in demonstration in the above two cases because the number of the scene images was one hundred for each.

Playback from different initial positions

Next we applied our method to pick-and-place, in which the initial positions of the object were different from those in demonstrations.

We performed nine demonstrations from initial positions shifted within $[\pm 0.1, \pm 0.1]$ in X- and Y-direction as shown in Figure 12. One hundred scene images were taken for each demonstration. Then trials of playback from one hundred random initial positions of the object shifted within $[\pm 0.1, \pm 0.1]$ in X- and Y-direction (Figure 12) were performed. The cumulative contribution ratio of the first one hundred principal components was 88% for the scene images in demonstration.

In this experiment, the success rate of the trials was 100%. Here "success" means that the object touches the goal circle in playback. This result means that our view-based teaching/playback achieved robustness against changes of task conditions: fluctuations of the initial position of the object. Offline computation time for this experiment is shown in Table 2.

Playback with noisy images in different lighting condition

In order to test the robustness of our method against image noise, random salt-and-pepper noise was added to the scene images both in the teaching and playback phases. The noise was added as follows:

$$I \leftarrow \begin{cases} 1\,(\text{white pixel}) & \text{when } r > 1 - \Theta \\ 0\,(\text{black pixel}) & \text{when } r < -(1 - \Theta) \\ I & \text{otherwise,} \end{cases} \qquad (6)$$

where I is the original gray-level of the pixel, r is a uniform random value in $[-1, 1]$ for the pixel, and Θ is a threshold. Figure 13 shows examples of scene images with noise when $\Theta = 0.01$ (1%) and 0.05 (5%).

Moreover, the lighting condition in the playback phase was changed from that in the teaching phase to make scene images brighter or darker. This was done by chang-

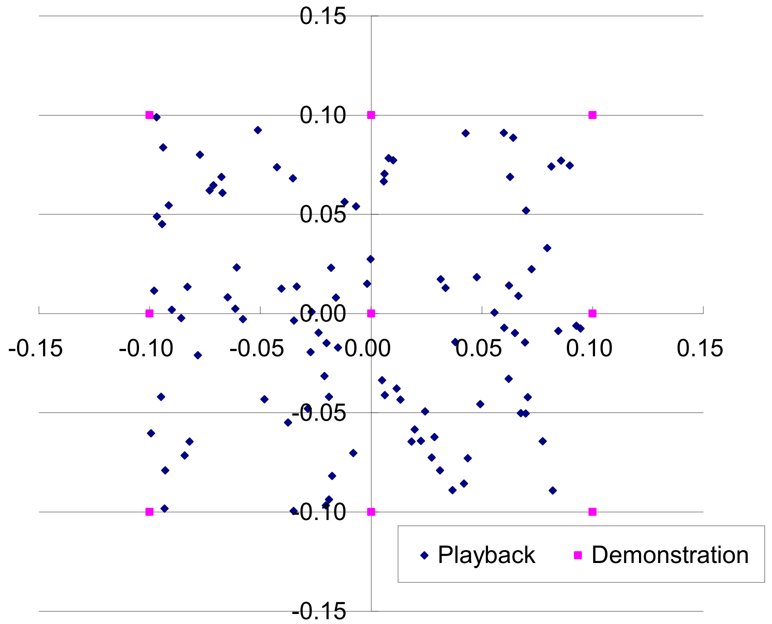


Figure 12 Initial object positions in demonstration and playback.

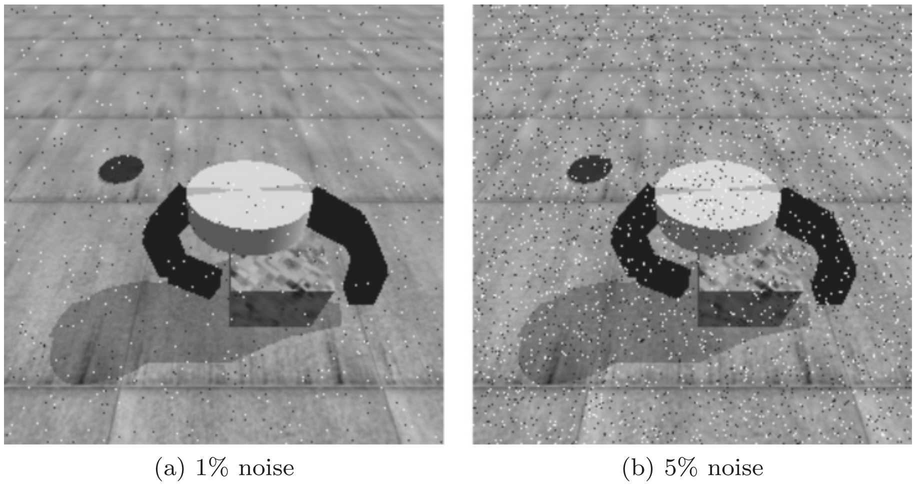


Figure 13 Example images with salt-and-pepper noise.

ing RGB values of the ambient light in OpenGL [19], which is used in ODE. Figure 14 shows examples of scene images in changed lighting conditions. The RGB values of the ambient light were changed from $(0.5, 0.5, 0.5)$ in the original condition to $(1.0, 1.0, 1.0)$ and $(0, 0, 0)$ in brighter and darker conditions, respectively.

We conducted teaching of pick-and-place in the same way as in the previous subsection. Nine demonstrations from different initial positions as shown in Figure 12 were performed, and one hundred scene images were taken for each demonstration. Then trials of playback from one hundred random initial positions of the object (Figure 12) were performed. The cumulative contribution ratio of the first one hundred principal components was 88% and 87% for the noise level of 1% and 5%, respectively.

The success rate of the view-based playback was shown in Table 3. The results show that our view-based teaching/playback can perform manipulation tasks even with image noise and change of lighting conditions to some extent.

Discussion

As shown in the previous section, our view-based teaching/playback enabled a robot hand to perform pushing and pick-and-place. Note that relative positioning of a hand with regard to a target as presented in [8] is unable to deal with such manipulation tasks.

In our view-based teaching/playback, camera calibration is unnecessary. The camera can be placed at any position from which the object and the end-effector of the

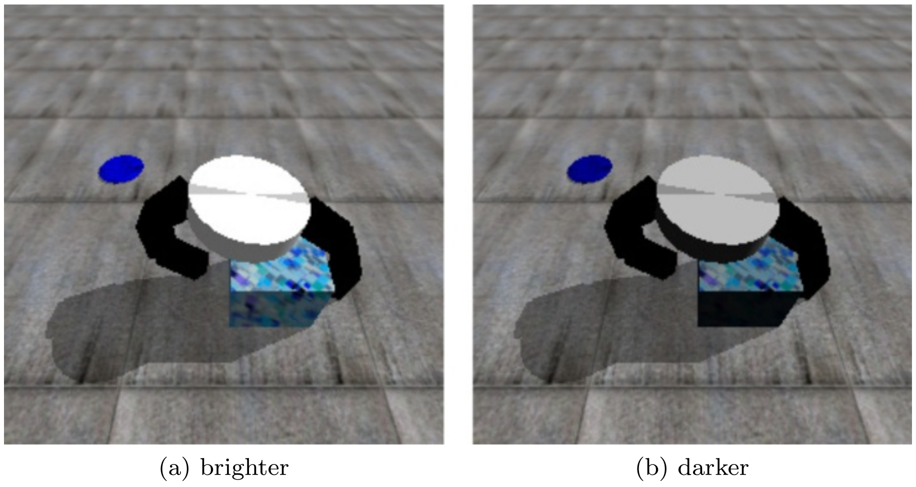


Figure 14 Example images in changed lighting conditions.

Table 3 Success rate of view-based manipulation

	Darker	Brighter
Noise (1%)	100%	100%
Noise (5%)	93%	84%

robot are observable. There is no need for careful tuning of lighting for vision. The view-based nature of our method leads to these advantages.

We can use two or more cameras easily in our method, if the increase of computational load is acceptable. Introduction of different types of sensor information, such as depth images and haptic images, is also straightforward. Information from such additional sensors can contribute to more reliable playback.

Of course, our method has some limitations:

- Because image information is mapped to robot movement by neural networks directly, it is difficult to teach jerky robot motion.
- Multiple human demonstrations must be "consistent". For example, let us consider pushing of an object to a goal avoiding an obstacle. If the object is carried on the right side of the obstacle in one demonstration and on the left side in another demonstration, a neural network would generate an interpolational robot motion to make the object hit the obstacle.
- Because our method is view-based, unexpected changes of camera views might disturb robot motions. We are investigating a scheme using an additional neural network to detect such changes [20] and a scheme with subimage-based neural networks to adapt to them [21].

In this paper, we do not use joint angles of the robot hand as the input to the neural network, because the information of the joint angles are included implicitly in $\mathrm{FS}(t)$. However, when the robot hand is often occluded, the joint angles should be fed into the neural network.

Conclusion

In this paper, we studied a new method for robot programming: view-based teaching/playback. It was developed to achieve more robustness against changes of task conditions, such as variations in initial positions of the object, than conventional teaching/playback without losing its general versatility. In this method, cumbersome camera calibration is not necessary because it is view-based.

The view-based teaching/playback was applied to pick-and-place and pushing in a virtual environment for proof of its concept. Using multiple demonstrations in the teaching phase, the virtual robot moved an object successfully to the goal position in the playback phase, even from some initial positions different from those in the demonstrations. The median filtering and gray-level normalization were used for robust view-based image processing against image noise and changes of lighting conditions.

Future work should address application of the proposed method to various manipulation tasks by actual robots. There is no need for special modification of our method for actual robots, and we have already applied it to simple pushing tasks by an industrial manipulator successfully [20,22].

However, further investigation is required to make our view-based teaching/playback more practical.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
YM proposed the method, designed the experiments, performed the analysis of the experimental results, and wrote the manuscript. TN implemented the method and conducted the experiments. All authors read and approved the final manuscript.

Acknowledgments
This work was supported by JSPS KAKENHI Grant Number 24560286.

Author details
[1]Faculty of Engineering, Yokohama National University, 79-5 Tokiwadai, Hodogaya-ku, 240-8501 Yokohama, Japan. [2]Nikon Corp., Tokyo, Japan.

Received: 14 August 2014 Accepted: 10 December 2014
Published online: 31 January 2015

References
1. Matsumoto Y, Inaba M, Inoue H (2003) View-based navigation using an omniview sequence in a corridor environment. Mach Vis Appl 14(2):121–128
2. Kuniyoshi Y, Inaba M, Inoue H (1994) Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Trans Robot Automation 10(6):799–822
3. Ikeuchi K, Suehiro T (1994) Toward an assembly plan from observation Part I: Task recognition with polyhedral objects. IEEE Trans Robot Automation 10(3):368–385
4. Ogawara K, Takamatsu J, Kimura H, Ikeuchi K (2003) Extraction of essential interactions through multiple observations of human demonstrations. IEEE Trans Ind Electron 50(4):667–675
5. Billard A, Calinon S, Dillman R, Schaal S (2008) Robot programming by demonstration. In: Siciliano B, Khatib O (eds). Springer Handbook of Robotics. Chap. 59.2. Springer, Berlin, pp 1371–1394
6. Argall BD, Chernovab S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Autonomous Syst 57(5):469–483
7. Zhao Q, Sun Z, Sun F, Zhu J (2008) Appearance-based robot visual servo via a wavelet neural network. Int J Control Automation Syst 6(4):607–612
8. Zhang J, Knoll A, Schmidt R (2000) A neuro-fuzzy control model for fine-positioning of manipulators. Robot Autonomous Syst 32(2-3):101–113
9. Shibata K, Iida M (2003) Acquisition of box pushing by direct-vision-based reinforcement learning. In: Proc. of SICE Annual Conf., SICE (The Society of Instrument and Control Engineers), Tokyo, Japan. pp 1378–1383
10. Kato M, Kobayashi Y, Hosoe S (2005) Optimizing resolution for feature extraction in robotic motion learning. In: Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics, IEEE, Piscataway, New Jersey. pp 1086–1091
11. Kobayashi Y, Kato M, Hosoe S (2007) Optimizing resolution for feature extraction in robotic motion learning. J Robot Soc Japan 25(5):770–778. (in Japanese)
12. Mason MT (2001) Mechanics of Robotic Manipulation. MIT Press, Cambridge

13. Kawabe T, Arai T, Maeda Y, Moriya T (2006) Map of color histograms for robot navigation. In: Arai T, Pfeifer R, Balch T, Yokoi H (eds). Intelligent Autonomous Systems 9. IOS Press, Amsterdam, pp 165–172
14. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev: Comput Stat 2(4):433–459
15. Turk M, Pentland A (1991) Eigenfaces for recognition. J Cogn Neurosci 3(1):71–86
16. Meireles MRG, Almeida PEM, Simões MG (2003) A comprehensive review for industrial applicability of artificial neural networks. IEEE Trans Ind Electron 50(3):585–601
17. Open Dynamics Engine. http://www.ode.org/
18. Craig JJ (2005) Introduction to Robotics: Mechanics and Control, 3rd edn. Pearson Prentice Hall, Upper Saddle River
19. Shreiner D, Woo M, Neider J, Davis T (2004) OpenGL Programming Guide. Addison-Wesley, Boston
20. Moriyama Y, Maeda Y (2013) View-based teaching/playback for manipulation by industrial robots. Trans Japan Soc Mech Eng Series C 79(806):3597–3698. (in Japanese)
21. Saito Y, Maeda Y (2013) View-based teaching/playback considering occlusion. In: Proc. of JSME Manufacturing Systems Division Conf., JSME (The Japan Society of Mechanical Engineers), Tokyo, Japan. pp 107–108. (in Japanese)
22. Maeda Y, Moriyama Y (2011) View-based teaching/playback for industrial manipulators. In: Proc. of IEEE Int. Conf. on Robotics and Automation, IEEE, Piscataway, New Jersey. pp 4306–4311