# INTERACTIVE 3D ANIMATION SYSTEM BASED ON TOUCH INTERFACE AND EFFICIENT CREATION TOOLS

*Anonymous ICME submission*

## ABSTRACT

Recently importance of tablet devices with touch interface increases significantly, because they attract not only mobile phone users, but also people of all generations including small kids to elderly people. One key technology on those devices is 3D graphics; some of them exceed the ability of desktop PCs. Although currently 3D graphics are only used for game applications, it also has a great potential for other purposes. For example, an interactive 3D animation system for tablet devices has been developed recently and attracts many people. With the system, multiple 3D animations are played depending on view direction and user's gesture input. However, the system still has two critical issues: the first one is a complicated process for data creation, and the other is a conflict on several gestures overlapped at the same position. In the paper, the first issue is solved by using a realtime range sensor and the second one is solved by considering the relationship between the 3D object and the input 2D gesture. The effectiveness of the system was proved by making the real system and creating and manipulating the actual 3D animations.

***Index Terms***— 3D animation, Touch interface, Realtime 3D scan

## 1. INTRODUCTION

Touch screen devices, such as iPhone, iPad and Android tablets become widely used and the touch interface becomes more common for ordinary people. In addition, the demand for touch interface is growing in all kind of personal computers, *e.g.*, Microsoft Windows 8 for PCs. On the other hand, the rich contents, such as videos and 3D computer graphics are expected to be killer contents for tablet devices. Although, videos become one of the most important contents for the devices, 3D contents are currently used only for games and not so actively used for other purposes. We consider that it is because the simple static 3D models or simple 3D video from single view point might not be enough attractive. If both contents are combined to allow users to interactively operate an animated 3D objects with touch interface, it will be widely applied to previously unseen contents.

Based on the fact, an interactive 3D animation system for tablet devices have been developed [2]. With the system, different 3D animations are played depending on a view direction to the 3D object and user's gesture input. However, the system has two critical issues to be more commonly used. The first one is the complicated process for data creation, and the other is a conflict on gesture recognition that comes from an ambiguity on retrieving 3D information from 2D input. In this paper, solutions for those two issues are proposed.

For data creation of 3D animation, mainly three processes are required; 1) creation of 3D motion data (we call the data "*3D animation*" in the rest of the paper), 2) definition of a state transition graph for 3D animations and 3) definition of gestures to call the corresponding animations to play. To make the process simple and easy, we propose a technique using an off-the-shelf realtime range sensor and efficient GUI.

The second issue is solved by a visual guidance on the 3D object considering the relationship between the 3D object and the input 2D gesture. Since 2D coordinate of the gesture is a projection of 3D object onto the touch screen, the coordinate changes dependent on the view direction, and thus, multiple gestures could be assigned at the same position on the screen. For solution, single gesture is selected by filtering the multiple gestures using depth information and the angle between view direction and the gesture direction. In addition, possible gestures are superimposed onto the 3D object to efficiently show the next possible gestures to the end users.

The remainder of the paper structure is as follows: Section 2 explains the related work and Section 3 is an overview of the proposed method. Section 4 and 5 detail the system of the method; Section 6 shows the experiments and a discussion. Finally, Section 7 concludes the paper.

## 2. RELATED WORK

Development of user-friendly contents is an important key to make the system popular to ordinary people [5]. In terms of 3D contents, tons of applications have been developed, however, only games made a success and few other applications are known. Recently, the framework of an interactive 3D animation was proposed [2]. It uses a state transition graph to represent the transition of an 3D object as a 3D animation such as "raise a hand", "open a tab," etc. One of the drawbacks of the system is that it requires several manual processes to make the contents, such as creation of 3D animations, construction of the graph and definition of the gestures to the graph. Further, there is no solution described how to

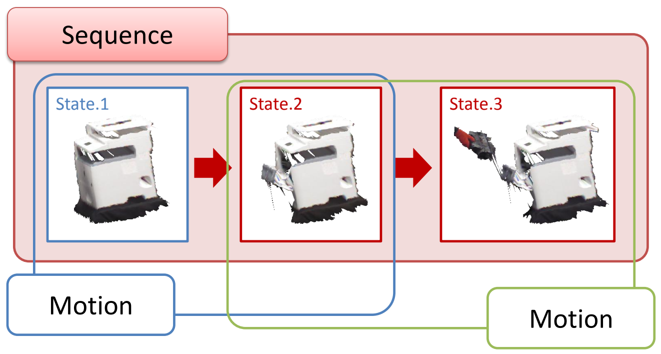


**Fig. 1**. Two types of animations: motion and sequence.

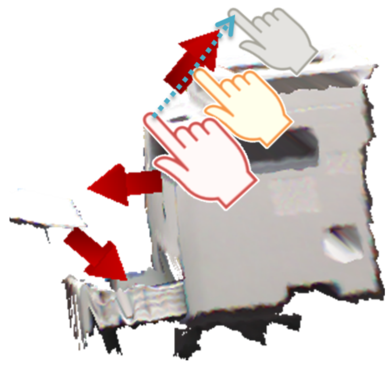


**Fig. 2**. Gesture for controlling animations.

solve the ambiguity between 2D gesture and 3D motion.

About gesture recognition, a number of frameworks for designing gestures are proposed [**?**][**?**]. Zhenyao *et al.* focused on the shape of a hand to define overlap-less gestures [**?**]. Chen *et al.* proposed the gestures for operate a 2D content instead of buttons [**?**]. Avrahami *et al.* propose a special (creator defined) curve by a finger to call a function [6]. To support a large number of functions, Wobbrock *et al.* propose a method detecting gestures based on the combination of the motions of fingers with a multi touch device [7]. However, all of the gestures are fixed on the 2D coordinate because the 2D contents do not change their shapes depending on the viewing direction.

Since the degree of freedom(DOF) of a 3D object is higher than 2D screen, special consideration is required to control the 3D object with 2D gesture. Cohé *et al.* proposed a recognition method of gestures for controlling 3D objects [**?**]. Bruno *et al.* proposed a design system of 3D objects by a touch interface and gestures [**?**]. In 3D animation case, one more degree is required to control the object for temporal direction (total 4D), and thus, previously proposed techniques cannot be simply applied.

## 3. OVERVIEW OF THE SYSTEM

The proposed system consists of two types of system; one is a animation creation system for contents creators and the other is an interactive 3D animation viewing system for end users.

The animation creation system helps the contents creator to compose short 3D animations into an interactive 3D animation with which the end user can interactively control the 3D animation. In this study, we define two types of animation, such as motion and sequence (Fig.1). The motion is an animation which represents simple action of the target, *e.g.*, "Open a box" or "Raise a hand." The sequence is a series of motions, such as "Take a box down from a shelf and open it." Before creating a sequence, the creator should make the 3D animations. With our method, 3D animations are automatically created by using range scanner. After that, the creator defines the state transition of the motions as a sequence. Then, the creator defines a gesture to call the sequence. This is one of the most important parts to create an interactive animation with touch interface. However, it is usually a complicated and difficult task to define the optimal gestures. Therefore, our system automatically detects the gesture from the captured data and also provides efficient GUI to edit the graph. Details of the system are described in Chapter 4.

We also implement the interactive 3D animation viewing system which works on Tablet PCs. The system detects the gestures of the end user who tries to manipulate the 3D animation and an object pose (Fig.2). Since the 3D object can be seen from arbitrary directions, 1) multiple gestures could be overlapped at the same position with similar direction on the screen, 2) some gestures are hidden by objects, or 3) some gestures are unstable if the directions of which are vertical to the screen. For solution, the proposed system automatically filters out the unsuitable gestures depending on the object pose and the viewing direction. Further guide signs for valid gestures are superimposed onto the 3D objects to help the end users to find possible animations. Details are described in Chapter 5.

## 4. ANIMATION CREATION SYSTEM

As described in the introduction, mainly three steps are required to make interactive 3D animation as follows; those are usually complicated and difficult tasks for non-professional users.

1. Creation of multiple 3D animations.
2. Defining a state transition graph for 3D animations.
3. Defining gestures to call the corresponding animations.

To make each process simple and easy to allow the contents creation to non-professional users, we propose the following three solutions. Details of each solution are described in the following sections, respectively.

1. Automatic creation of 3D animation by making the sequence of 3D polygons from a point cloud data which is acquired by the Kinect [3, 4].

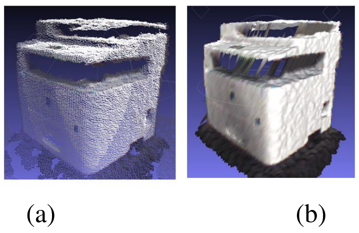


**Fig. 3**. Scanned by Kinect. (a) raw data of point cloud and (b) the created polygon for animation.

2. Efficient GUI system for defining the state transition graph.

3. Automatic gesture assigning method by analyzing the operator's hand motion from the captured data.

## 4.1. Scanning 3D motions by using Kinect

In our interactive 3D animation system, a motion consists of a series of static 3D polygons. Usually, a 3D animation is represented by deformations and transitions of a polygon. However, the creation process of this kind of data is complex and required special knowledge of 3D animation. Therefore, we use Kinect for scanning 3D motion of an object to realize the easy creation of 3D animations. Since raw output from Kinect is a 3D point cloud of the target scene as shown in Fig.3 (a) and the point cloud is not suitable for rendering the 3D animations, our system automatically creates polygon surfaces form the point cloud as shown in Fig.3 (b). Even if the size of this data is larger than the animation data which consists of a polygon with its deformations, we consider that our approach is currently the only solution for practical usage because the latter data cannot be created with the state-of-the-art technique yet [**?**]. Once 3D animations are created, a state transition graph of those motions for representing a sequence is created.

## 4.2. Definition of a state transition graph

We need a state transition graphs for representing a 3D animations. Each edge of the graph represents physical transitions of motions which correspond to 3D animation as shown in Fig.4. In our system, there are two animations such as motion and sequence. Since a sequence consists of a series of motions, definition of the sequence is an important task. To make definition simple, only start and end node are required in our method and the system automatically finds the path from the start to the end as shown in Fig.4 bold arrows.

We also implement a graphical user interface for contents creators to create motion graphs easily. This tool enables the creators to define the relationships of the motions with icon images. Fig.5 shows the example of the GUI while defining the motions for the multifunction printer. In this example, we define five edges as gray lines in the figure out of possible 30

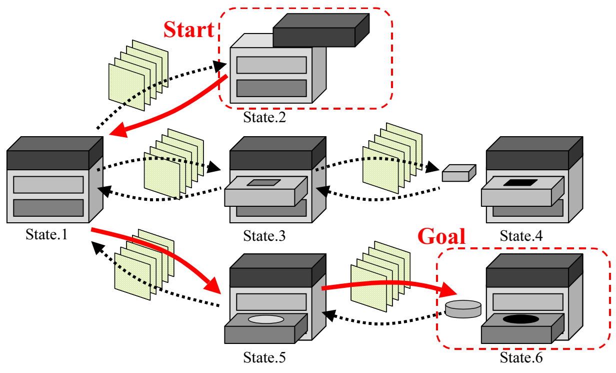
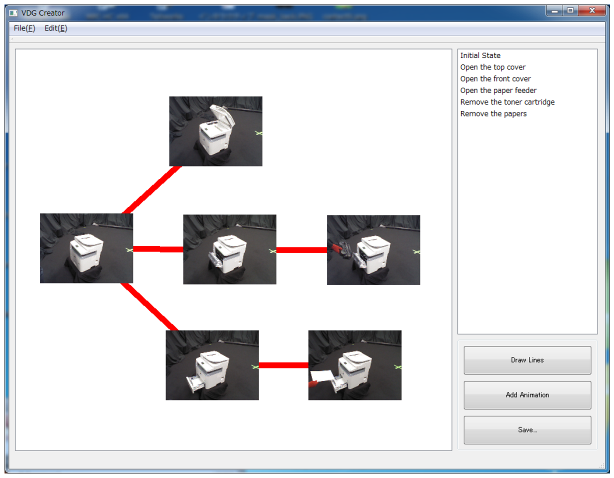


**Fig. 4**. Graph representation of a state transition model. Edge (arrow in the figure) corresponds to a specific animation. Once start and end nodes are given, a sequential animation is automatically created as a shortest path (bold arrows).



Initial State
Open the top cover
Open the front cover
Open the paper feeder
Remove the toner cartridge
Remove the papers

**Fig. 5**. Graphical user interface of the animation creation system for creating a motion graph.

edges which represents all the transitions from every state to all others.

## 4.3. Automatic detection and assignment of gesture

### *4.3.1. Automatic detection of the movement of a hand*

Manual detection of the gesture for each 3D animation is a complicated and laborious task. To solve the problem, we propose an automatic technique to detect gesture from 3D animation by image and geometry processing. The key idea of the technique is to use the hand in the 3D animation which is captured at the same time while scanning the target object; note that the intuitive gesture for manipulating an object with touch interface is basically as same as the movement of an operator's hand in real world. The proposed system analyzes the movement of an operator's hand by using a RGB image and depth image given by Kinect as shown in Fig.6. The actual hand motion detection processes are as follows:

1. Detect the area of color of a hand from a RGB image.

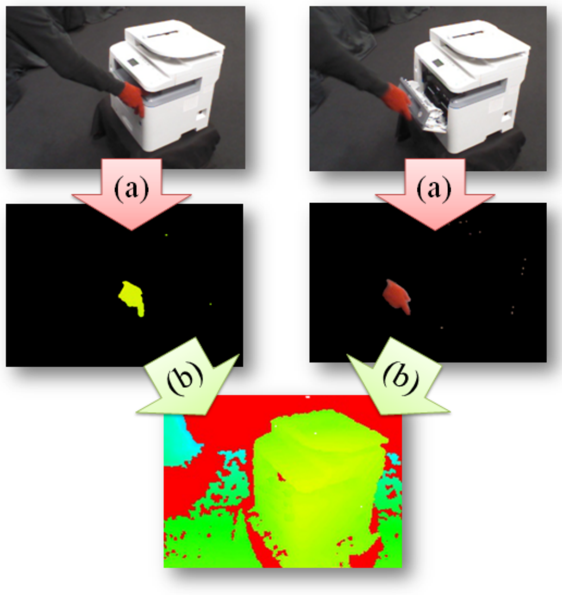


**Fig. 6**. Detection of a hand motion: (a) Detecting an area of a hand in RGB images. (b) Detecting a collision between the hand and an object by using a depth image.

2. Calculate the center of the hand area avoiding the outliers in each frame by using mean-shift algorithm [**?**].
3. An originally captured sequence contains unnecessary motion of the hand; *e.g.*, when capturing the sequence of "Open the box" motion, hands moves freely before and after opening the box. To remove those unnecessary motion, we use the depth information to detect whether the hand is touching the target object or not, and only the hand motion touching the object is used. With this process, both the starting and end positions of the motion and the direction of it are retrieved.

### 4.3.2. *Definition of gesture from the detected hand motion*

Since the detected motions of the hand in the previous section are continuous values, we calculates the simple motion by averaging them. Since we quantize the angle of the direction of the gesture into eight directions to avoid mis-manipulation, we approximate the angle of the direction in steps of 45 degrees (Fig.7). If a conflict between several gestures is found by comparing both the starting positions and the directions of the motions, we change the direction to the next nearest angle. If the conflict cannot be solved by the operation, we simply remove the gesture.

### 4.3.3. *Automatic assignment of gestures on graph*

With our system, one gesture corresponds to the one 3D animation, and the 3D animation is linked to the specific edge of the state graph as shown in Fig.8 (i). In other words, these relationships are all one-to-one and no duplication exists. Considering the fact that each gesture is attached to the 3D object, we assign all the gestures to all nodes of the graph as shown in Fig.8 (ii); note that since 3D animation is reversible, the direction of the gesture flips at the edge where the corresponding animation is linked.[1] By the process, gestures are automatically assigned to the graph.

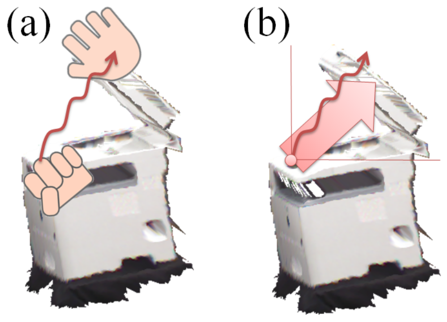


**Fig. 7**. Detected motion and the gesture.

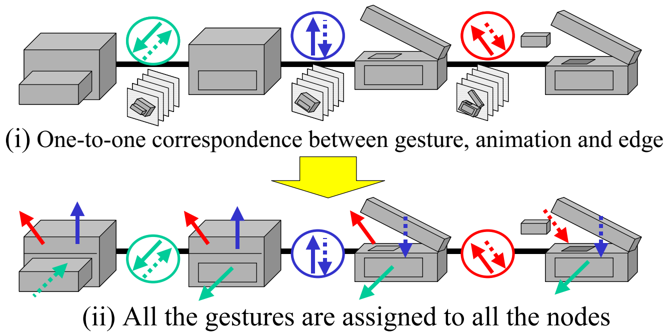

(i) One-to-one correspondence between gesture, animation and edge
(ii) All the gestures are assigned to all the nodes

**Fig. 8**. Example of automatic assignment of gestures on the state graph. Each arrow represents the gesture. Note that since 3D animation is reversible, the direction of the arrow flips at the edge where the corresponding animation is linked.

## 5. ANIMATION VIEWING SYSTEM

### 5.1. View dependent gesture with touch interface

One of the important feature of the 3D animation viewing system is that it needs to change the position and the direction of the gestures depending on the view direction and the poses of 3D objects as shown in Fig.9. For solution, the system calculates both the distance and angle in 2D space by projecting 3D motion into 2D touch screen to find the closest gesture.

Actual process is as follows. First, 3D motion which are consistent with the starting position $Sg(x, y, z)$ and direction $Dg(x, y, z)$ are projected to the screen coordinate as $Sg'$ and $Dg'$. Input data are given by the tracks of fingers on touch interface, *i.e.*, the system detects the starting position $St(x, y)$ and direction $Dt(x, y)$ on the screen coordinate. Then, the system calculates both the distance between $St$ and $Sg'$, and the inner product between $Dt$ and $Dg'$.

[1] currently loop is not assumed for the state graph.

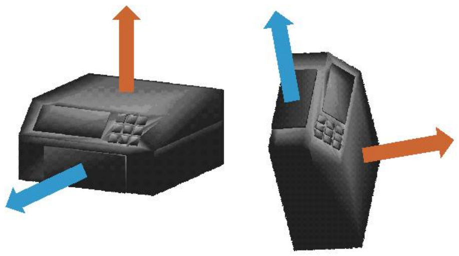


**Fig. 9**. Position and direction of the gesture change automatically dependent on the relationship between the view direction and the object pose.

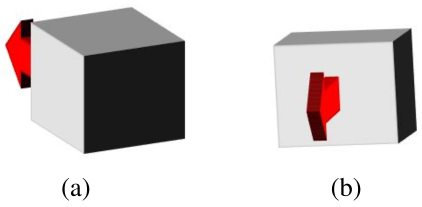


**Fig. 10**. (a) Starting points of gestures are hidden by objects and (b) the direction of the gestures becomes unstable because the directions of the gesture is almost vertical to the screen.

## 5.2. Gesture filtering and visualization for user guide

Since the 3D animation can be seen from an arbitrary direction with our system, the starting points of gestures are sometimes hidden by objects as shown in Fig.10 (a) or the direction of the gestures becomes unstable if the directions of which are vertical to the screen as shown in Fig.10 (b). Therefore, the system disables the gesture if the starting point of the gesture is hidden by the object or the angle between the gesture and the viewing direction is steeper than a certain degree. Then, the system visualizes the starting positions and directions of only valid gestures by arrow signs. With the signs, the end users who are not familiar with the system can easily understand possible gestures to play the animations.

## 6. IMPLEMENTATION AND DISCUSSION

To confirm the effectiveness of our system, we create interactive 3D animation using Kinect. We create an electric manual for a multifunction printer. The Data acquisition environment is shown in Fig.11.

## 6.1. Creation of a multifunction printer animation

We scanned five animations (motions) of the multifunction printer. The details of each animation are as follows:

1. Open the top cover to use the scanning function.
2. Open the front cover to access the internal parts.
3. Open the paper feeder.
4. Remove the toner cartridge.
5. Remove the papers.

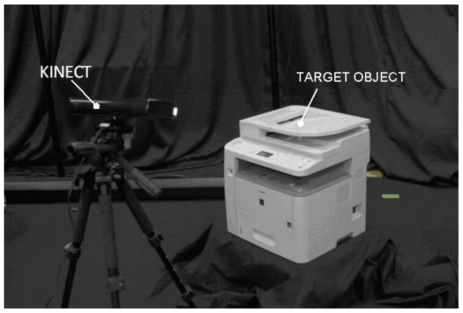


**Fig. 11**. 3D animations capturing scene.

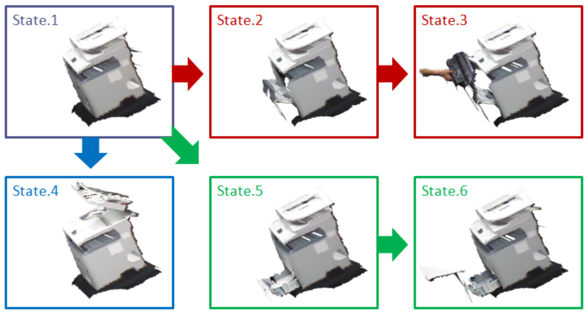


**Fig. 12**. State transition graph of the multifunction printer.

As shown in Fig.12, 3D animations which consist of sequential polygon data from the point clouds were successfully created by our method. Then we configured the state graph with six nodes (states) using the GUI. The graph was successfully constructed as shown in Fig.12.

## 6.2. Automatic detection and definition of gestures

Then, we tested the automatic detection of gesture from the captured 3D animations. To detect the hand of the operator easily, the operator wore a red grove. Fig.13 shows the result of the automatic detection by our method. The multifunction printer has the toner cartridge and the paper tray inside the front cover. The result shows that our system correctly detects the position and direction of the front cover, the toner cartridge and the top cover as the starting point of the gesture.

We also confirmed that the 3D animation viewing system automatically selected only the valid gestures depending on the viewing direction. As shown in Fig.14 (b), there were originally nine gestures on the 3D object including some of them have the direction vertical to the screen. However, only valid gestures remained with our filtering technique as shown in Fig.14 (a).

## 6.3. Implementation on the tablet PC

We implement our system on the tablet PC, Eee Pad TF201, Android Ver. 4.0.3. The rendering API is OpenGL ES [8]. This system can play each animation in real time as shown in Fig.15. However, since our 3D animation consists of a large number of polygons, it takes about two minutes to read the polygon at the startup time. This is our important future work to reduce the polygon data to solve the problem.

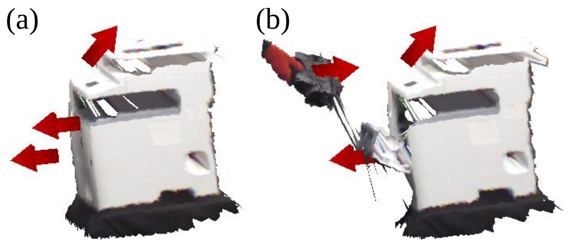


**Fig. 13**. Results of the Automatic detection of gestures. (a) Gestures around the front cover at the initial state. (b) Gestures after the front cover is opened (state 3 in Fig.12).

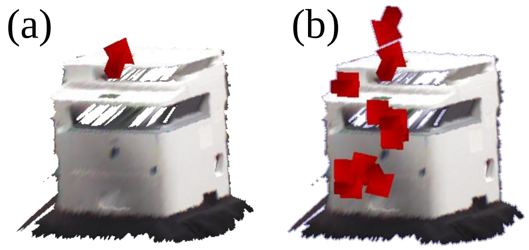


**Fig. 14**. Selection of the controllable gestures. (a) With the selection and (b) without the selection. In (b), most of the gestures are overlapped and directions are vertical to the screen, however, those are automatically removed in (a). If the user changes the object pose, those gestures appear again.

## 7. CONCLUSIONS

We proposed the 3D animation creation system that enables non-professional users to create the interactive 3D animations easily to promote the 3D contents for tablet devices. We also proposed the animation viewing system to control 3D animation by gestures. The contributions of this study are summarized as follows:

1. Automatic creation of 3D animations using an off-the-shelf realtime range scanner is realized.
2. Automatic detection of gesture and assignment to the corresponding edge on the state graph using captured data are realized.
3. Automatic selection of valid gestures with efficient visualization technique is realized.

For future direction, extension to an entire shape of 3D animation with reduced number of polygon should be important.

## Acknowledgment

This work was supported in part by SCOPE No.101710002 and NEXT program No.LR030 in Japan.

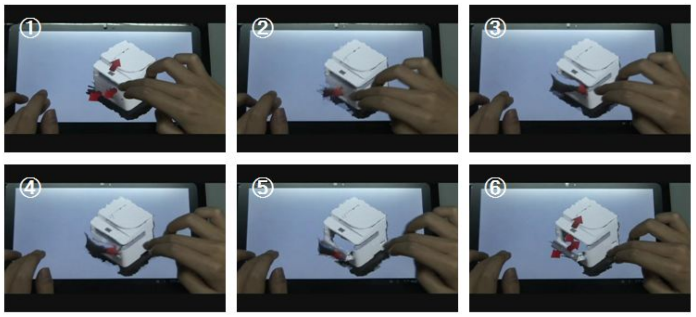


**Fig. 15**. Manipulating animation with touch interface.

## 8. REFERENCES

[1] Masahiro Ishikawa, Hiroshi Kawasaki, Ryo Furukawa, and Yukiko Kawai, "Shape rank: efficient web3d search technique using 3d features," in *Proceedings of the international conference on Multimedia information retrieval*, New York, NY, USA, 2010, MIR '10, pp. 393–396, ACM.

[2] M. Furukawa, S. Fukumoto, H. Kawasaki, and Y. Kawai, "Interactive 3d animation system for web3d," in *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, july 2012, p. 666.

[3] Zhengyou Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4 –10, feb. 2012.

[4] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox, "Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments," in *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[5] Yukiko Kawai, Shogo Tazawa, Ryo Furukawa, and Hiroshi Kawasaki, "Efficient meta-information annotation and view-dependent representation system for 3d objects on the web," in *ICPR'08*, 2008, pp. 1–4.

[6] Daniel Avrahami, Scott E. Hudson, Thomas P. Moran, and Brian D. Williams, "Guided gesture support in the paper pda," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*, New York, NY, USA, 2001, UIST '01, pp. 197–198, ACM.

[7] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2009, CHI '09, pp. 1083–1092, ACM.

[8] P. Cole, "Opengl es sc - open standard embedded graphics api for safety critical applications," in *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*, oct.-3 nov. 2005, vol. 2, p. 8 pp. Vol. 2.