Touch-RE: A Touch-based Model Acquiring Method for Personal Manufacturing

Jinho Jung Yong Chen*
Epstein Department of Industrial and Systems Engineering
University of Southern California, Los Angeles, CA 90089

*Corresponding author: yongchen@usc.edu, (213) 740-7829

ABSTRACT

Enabled by additive manufacturing technology, personal manufacturing, if realized, will give ordinary people control over their physical world by allowing them to personally program its construction. Currently several pioneering mass customization and personal fabrication applications have been developed for consumer products, gifts, toy, etc. However, one of the core challenges that are remained to be addressed for realizing personal manufacturing is the easiness of creating 3D models for additive manufacturing systems. Current developments in CAD software and reverse engineering are still lacking on the easiness of usage especially for normal users without much experience. In this paper, we present a touch-based 3D-shape acquiring method that is easy and intuitive to use. The key technical challenges on the related hardware and software development are discussed. A testbed system is presented and multiple examples are given to demonstrate its capability.

Keywords: Reverse engineering, Wiimote CAD, 3D modeling, augmented reality.

1. INTRODUCTION

Enabled by digital devices, personal communications and computation have revolutionized our lives by allowing us control over our personal information. *Personal manufacturing*, if realized, will likewise give ordinary people control over their physical world by allowing them to personally program its construction. Currently pioneering *mass customization* and *personal fabrication* applications have been developed for consumer products [1], collectables and gifts [2], toys [3], and game avatars [4]. Recent advances in material, process and machine development have also enabled the *solid freeform fabrication* (SFF) processes to evolve from high-end prototyping to low-cost 3-Dimensional (3D) printing. Nowadays there have been commercial 3D printers that are priced under $10K. We expect future developments on 3D printing will make the technology more widely available, and eventually be adopted for personal usage.

Even with low-cost 3D printers available, however, a core challenge to be addressed for the realization of *personal manufacturing* is the easiness of creating computer-aided design (CAD) models that are customized to personal needs. That is, the CAD model creation process needs to be extremely easy even for the masses who may have no technical knowledge on CAD and CAM. In addition, such model creation system needs to be low cost in order to successfully address the mass market. In this paper, we will focus on how to create a digital model from a physical object for the purpose of duplicating, repairing, and redesigning the object.

Reverse Engineering (RE), by converting an existing physical object into a digital model, has been well developed and widely used [5]. Various technologies have been developed before including coordinate measuring machines (CMM), laser scanner, structured light digitizers, and computed tomography. As shown in Figure 1, Reverse Engineering typically involves the following process. A physical object is first scanned into point cloud data; polygonal surfaces are then constructed from the scanned points. For the purpose of part repair or model re-design, further processing of the data is required in order to generate a feature-based CAD model. Typically the process involves NURBS reconstruction and feature extraction modules. Hence the current usage of reverse engineering is lengthy and would be hard to be adopted by *personal manufacturing*.

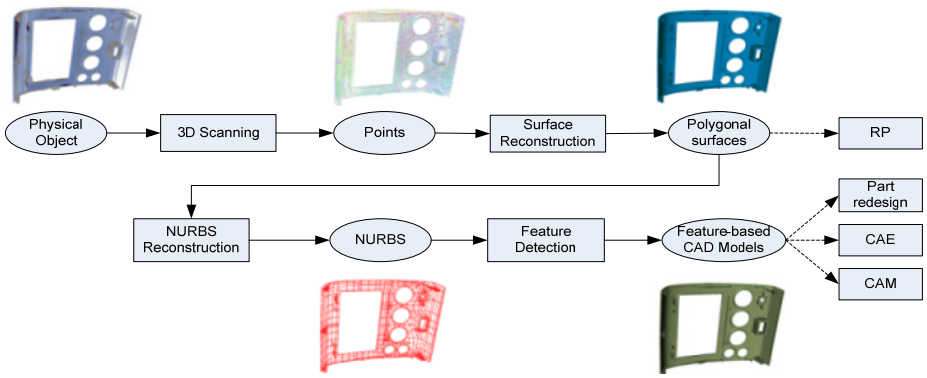


Figure 1: Typical data flow of Reverse Engineering.

In this paper, we present an alternative way of acquiring digital models from physical objects. Motivated by the popularity of *IPhone* and *IPad* developed by *Apple Inc.*, which are all touch based systems, we try to integrate touching input in a CAD system to simplify Reverse Engineering. Our goal is to develop an inexpensive reverse engineering system (called touch-RE) that is easy to use by all types of users. Hence it would enable the masses to better use their 3D printers, if available, to duplicate, repair, or re-design their physical objects. The data flow of our method, in comparison, is shown in Figure 2.

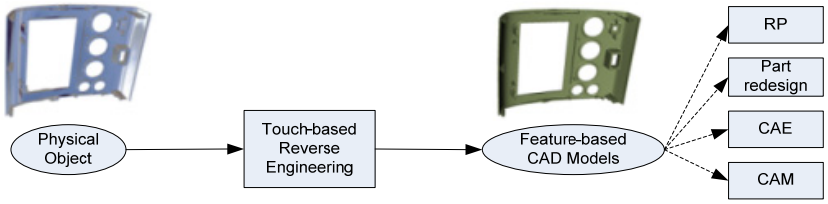


Figure 2: Data flow of Reverse Engineering.

2. OUR APPROACH – TOUCH-BASED REVERSE ENGINEERING (TOUCH-RE)

In our daily life, touch and vision are co-located during the exploration of objects. The perception of object shapes is guided by vision. Accordingly it is intuitive for the user to touch the surfaces of the recognized shapes. For example, as shown in Figure 3, the object in the real world is easily to be recognized as a cylinder. Hence the user can touch a minimum of three points (P_1, P_2, P_3) on the cylindrical surface to uniquely define the cylinder in the space. If the touching points (x_i, y_i, z_i) can be accurately recorded, a cylinder model in the virtual world can be easily computed. The computed CAD model can then be displayed to the user as visual feedback. In addition, since the cylinders in the real and virtual world have a known correspondence, we can keep them synchronized. That is, if we rotate the physical object along Z axis by an angle α, we can update the virtual object based on the same rotation, and vice versa.

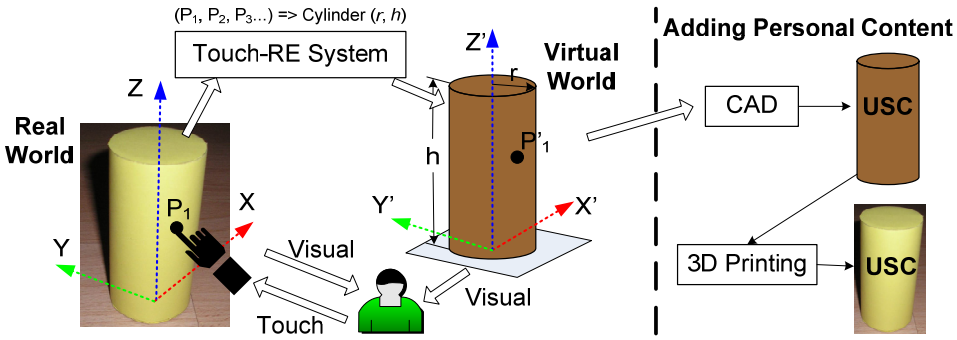


Figure 3: Illustration of the principle of Touch-RE.

Hence the core idea of touch-RE is to allow the user to feel and see the shape of real world and use the user's touching input to create the shape of virtual objects in a co-located way. Traditional computer-aided design (CAD) systems require the user to visualize the shape and use standard input devices such as mouse and keyboard to define it. In comparison, touch-RE allows the user to look at the real object instead of screen and visualize the virtual object side by side with the real object. Hence, 3-D indication and visualization become trivially natural and intuitive, which is critical for the mass market. The created feature-based virtual object can be used for various purposes. For example, personal content can be added and the modified 3D model can then be built by using a 3D printer.

As discussed in [6], the real world and a totally virtual environment are at the two ends of a continuum with the middle region called Mixed Reality. Traditional CAD systems are based on virtual reality, that is, both the input and display are based on a totally virtual environment. Augmented reality (AR) or augmented virtuality (AV) lies between the real world and a totally virtual environment. There have been extensive researches on both AR and AV. As discussed in [7], the goal of AR systems is to superimpose (i.e., augment) computer-generated graphics over real objects in the user's field of view as if they appear in the real world. Most work in AR uses head-mounted displays (HMD) to bring the (see through) display to the working field [8~10]. In the other side, augmented virtuality refers to predominantly virtual spaces, where physical elements such as physical objects or people can be integrated into [11].

In our method, we adopt the principle of AR by integrating real and virtual objects together. Accordingly a reverse engineering system called **Touch-RE** has been developed, which can assist users to rapidly acquire 3D models of physical objects. Unlike other RE systems, real object's information (shape, size and position) can directly be captured in our system. Figure 4 presents a simple example to demonstrate how our system works. In the example, three touch motions (left-bottom and right-bottom for base circle's radius and position, and left-top for its height) enable the Touch-RE system to generate a cylinder shape including its sizes and position. The whole process only takes several seconds.

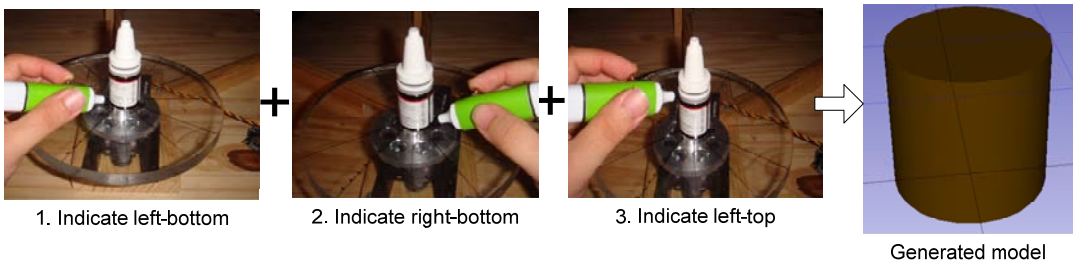


Figure 4: Acquiring the shape of a cylinder.

Since our system is mainly to create 3D shapes based on user's touching input, a main research issue to be addressed is how to integrate touches in a RE system for defining general shapes. It is desired for the user to provide touching input easily, quickly, and accurately. In addition, since the acquired touching point location will be used in computing virtual models, it is desired to have a small error from $\boldsymbol{P_i}$ in the real world to $\boldsymbol{P_i}'$ in the virtual world. Hence the computed virtual model would be accurate. At the same time, the system should be low cost. We investigated both hardware and software issues for addressing the identified requirements. Details of our method are presented in the rest of the paper, which is organized as follows. Section 3 describes the selection of hardware components and their setup in the system. Section 4 presents a 3D coordinate computation method from multiple 2D Infrared cameras. Section 5 describes the data flow of our system and related software constitution. Section 6 explains the model generation and alignment methods. Section 7 presents two examples to illustrate the systems' capability. Finally Section 8 concludes the paper.

3. HARDWARE OF TOUCH-RE SYSTEM

In the section, we introduce our design decision on hardware selection and system layout for integrating touches in defining general shapes.

3.1. Technology Selection

Various touching devices based on different technologies have been developed before.

(1) 3D measuring arm such as FAROARM by *FARO Technologies Inc.* are widely used in Reverse Engineering. It is a technology based on inverse kinematics. A set of joints are used to link a touching probe to achieve desired freedom. By measuring current angles of joints, the 3D coordinates of the contact point can be accurately computed (e.g. 0.02mm accuracy). However 3D measuring arm is expensive with a price range of $10K ~ $100K.

(2) Haptic devices such as PHANTOM Omni device from *Sensable Technologies* can provide 6 degree of freedom tracking and force feedback. They have been widely used in interactive 3D carving and sketching. However, the motion region is usually limited.

(3) Based on triangulation method, two cameras can capture the intersection point of an object surface and a ray shot by a laser. This approach has also been widely used in Reverse Engineering. However, it is very difficult for a user to accurately control a laser from a distance to indicate a touching point.

(4) Infra Red (IR) LED and related IR-sensor is a method that has recently becoming popular in human-computer interaction. One of the cheapest IR signal receiver is a Wii controller from *Nintendo* as a game input device. Lee [12] suggested a novel way of hooking Wii controller's Bluetooth signal with personal computer (PC). From that research, he provided a 2D drawing application based on one Wii controller and one IR-LED. Based on it, Hay et al [13] showed that two Wii controllers can be used for getting a 3D point's coordinate with good accuracy (average error at 2.46mm with a standard deviation of 2.23mm).

A comparison of the described technologies is given in Table 1.

Table 1: Comparison of different technologies (highlighted sections in blue indicate dominate characteristics)

Properties	3D arm	Haptic	Laser & Camera	IR LED & Sensor	Decision priority
Total cost	High	Medium	**Low**	**Low**	1
Accuracy	**High**	**High**	Low	Moderate	2
Hand freedom	Low	Low	**High**	**High**	3
Device size	Big	Medium	**Small**	**Small**	4
Operation region	**Large**	Small	Medium	Medium	5
# of working points	1	1	1	**4**	6
Calibration needs	**No**	**No**	Yes	Yes	7

As discuss before, it is desired in our system that a user can provide touching input easily, quickly, and accurately. In addition, the touching device needs to be low cost, light weight, and easy to move. Accordingly our technical priority in comparing the technologies is shown in Table 1. Based on the comparison, we decided to use IR LED and sensor in our system. It has good properties such as its low cost, easy for hand to move, small device size, and reasonable accuracy. We will discuss how to achieve good accuracy based on Wii controllers in Section 4.

3.2. Hardware Development

Based on the well-known triangulation method, a 3D point coordinate can be computed based on the input of two Wii controllers. However, a user may block some views when interacting with a real object. Hence, instead of constraining user's touching motion to prevent such blocking, we decided to use four Wii controllers that are mounted from various angles. Further efforts are made to ensure at least two Wii controllers can provide input in computing 3D coordinates of touching points. Due to the unavoidable blocking by an object itself, we decided to mount all the four Wii controllers in one side of the object. To provide an all round view of the object, a rotation table has been integrated in our system.

The operation region of a commercial Wii controller is examined first. Based on our experiments, a Wii controller could sense a maximum of 745mm × 580mm rectangle area when the target is 1000mm

away. This means that a Wii controller senses horizontally 40.88° and vertically 32.36°. Since a Wii controller returns more accurate coordinate information for a closer distance, we will put Wii controllers as close as possible to the working volume as long as they can provide sufficient coverage of the volume. In our development of the Touch-RE system, the maximum size of a real object that we decided to accept is a 100mm × 100mm × 100mm cubic shape. Accordingly the Wii controllers are put away ~310mm from the center of the working volume to provide sufficient margins (~35mm) around the volume.

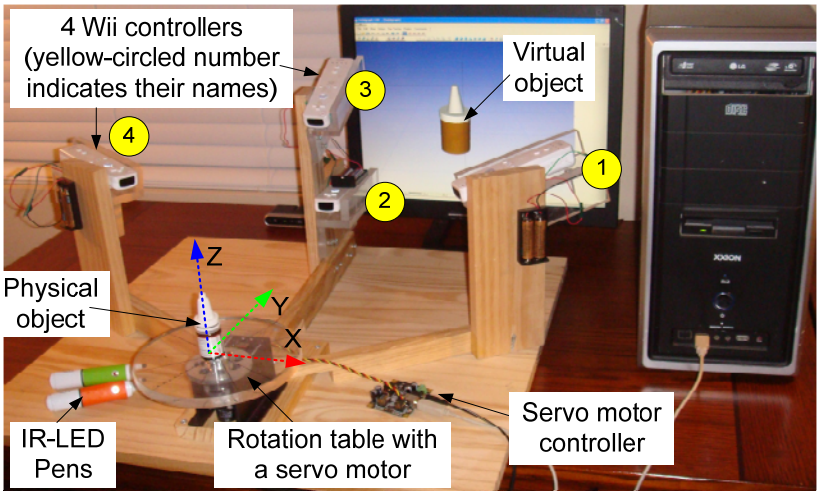


Figure 5: Setup of Touch-RE system.

The designed hardware setup of our Touch-RE prototyping system is shown in Figure 5. The three major hardware components of the system are explained as follows.

(1) Wii controllers: Wii controllers have been developed for game market. They are widely available at rather low price (~$25). The device has a built-in infra-red sensor that can detect up to 4 different IR-LED signals. Every time a Wii controller senses IR signals, it detects their positions and radiates Bluetooth signal. If a PC has a Bluetooth receiver, the signal from the Wii controller can be recorded and further decoded by using a Wiimote library. Currently C, C++, and C# version of Wiimote libraries are widely available. The decoded 2D position has coordinate ranges of 0 ~ 1023 in *X*-axis and 0 ~ 767 in *Y*-axis.

(2) Rotation table with a servo motor and driver: One problem of our system is that we can only touch objects from one side. By using a rotation table, we can rotate with a precise angle to enable touching from another side. For example, if we want to touch a portion of an object that is being blocked by itself from the Wii controllers, we can rotate the object along the *Z*-axis by 180 degrees. Since the rotation information has been maintained, the system knows that the next input, compared to the previous one, has been rotated 180 degrees along *Z*-axis. For acquiring repeatability, we used a HS-985MG servo motor, and a Phidget's AdvancedServo 1061 driver for controlling the servo motor. A rotation table is mounted on top of the servo motor with a set of gears to achieve sufficient torque.

(3) IR-LED pen: A IR-LED pen has a 940nm wavelength infrared LED on its tip. The LED has a viewing angle of ±25°. The pen has a switch on its body; hence we can easily turn on/off the LED as needed. The real location of the pen that is identified by a Wii controller is the 'hot-spot', the place with the strongest light intensity. Due to the viewing angle, such a position is around the touching point of the IR-LED and the object. Two IR-LED pens are used to enable two-hand actions. Since a Wii controller can sense up to 4 IR signals, two more IR-LED pens can be added in the system to enable potential collaboration of multiple users.

3.3. Hardware Verification

Based on the prototype structure, we examined the operation region and the coverage of Wii controllers. As shown in Figure 6, Wii #1 and #4 are used for covering the left and right side of the object

respectively; while Wii #2 and #3 are used to track touching on the top and bottom side of the object. We used a cup as a test example. When touching it in the front-middle view, all the 4 Wii controllers can sense the location of the IR-LED. For the rest cases, IR-LED points can be sensed by at least 3 Wii controllers. To ensure our application can cover 100mm cubic space, we also used a cube with such size and touched 8 vertices of the cube and checked how many Wii controllers can sense them. The results indicate that all the 8 vertices can be sensed by at least 3 controllers.

(a) Right view (b) Top view (c) Bottom view (d) Left view

Figure 6: Different views and region of operations.

4. ACQUIRING 3D COORDINATE OF A TOUCHING POINT

Every time a Wii controller senses an IR light, it will send Bluetooth signal to our application. We can decode the signal and get the 2D coordinate value (x, y) in its local coordination system. No signal will be sent by the Wii controller if no IR light is presented in its viewing field. Hence, when a user switches on an IR-LED pen to indicate a touching point, depending on the viewing angle, our applications will get all or some of 2D points (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) from the 4 Wii controllers respectively. Hence a key problem to be addressed is how to compute a 3D coordinate (x, y, z) from the received 2D points in order to accurately define the touching point.

4.1. 3D Coordination Acquiring Principle

Essentially a Wii controller is an IR camera. It is necessary to know the focal point of each camera in order to perform triangulation. There have been extensive researches on camera calibration. For example, Zhang [14] introduced a camera calibration method for calibrating a camera easily. And Hay *et al.* [13] implemented a Wii-based 3D pointing application by using Zhang's method and camera calibration toolbox. To use Zhang's method, we need to know the relationship of two cameras (extrinsic matrix) and each camera's characteristics (intrinsic matrix). For calculating such information, a square-shape IR-LED array can be made to generate sufficient samples from various positions and orientations. Hence a camera calibration toolbox can be used to compute the intrinsic and extrinsic matrices. Based on these matrices, triangulation method can be used to transform two 2D points into one 3D coordinate.

Even though this method is well established and can achieve results with reasonable accuracy (e.g. average error at 2.46mm with a standard deviation of 2.23mm in [13]), it is a general method without considering the unique properties of our system such as:

(1) In our system, each controller can accept signal and send its 2D coordinate to our application. Hence we usually receive inputs from 3 or 4 Wii controllers. However, the aforementioned method only considered 2 calibrated cameras. For the input of 3 or more cameras, even though we can compute results of each pair of camera and use their average value, it is desired to have a method that can achieve better accuracy by using the input from more cameras at the same time;

(2) The position and orientation of all 4 Wii controllers are pre-defined and fixed. Hence their calibration only needs to be done once. We would prefer a calibration method that is more accurate even it takes longer time and efforts.

Hence, we developed a 3D coordinate computation method based on machine learning. The test results indicate that our method can achieve a better accuracy inside the working volume of the system.

4.2. Our Acquiring Method based on Machine Learning

The basic idea of our method is straightforward. It contains two major steps:

(1) Build an extensive database on the relations between pre-defined sampling points (x, y, z) and related 2D points (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) from Wii controllers;
(2) Based on the built database and identified patterns, compute an unknown 3D coordinate (x, y, z) from any given Wii controllers' input, i.e. all or portion of (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4).

The two steps are discussed in more details as follows.

(1) Building a database by sampling the working volume

As discussed in the hardware development, the working volume of our system is a 100mm × 100mm × 100mm cubic. All the touching input can only happen in this volume. Hence we can use a 3-axis linear slides (accuracy: 0.076mm) to move an IR LED inside the volume. Then we can record 2D points (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) of 4 Wii controllers for a known 3D coordinate (x, y, z) based on the motion of the linear slides. The hardware setup is shown in Figure 7. We sampled the working volume by an increment of 5mm in X, Y, and Z axis. Hence a total 8000 (20×20×20) samples has been taken. For the validation purpose, we also took another 1000 (10×10×10) samples by shifting the IR-LED 2.5mm in X, Y, and Z axis and then sampling the volume by an increment of 10mm. Table 2 shows some examples of the collected calibration data. One drawback of this approach is that it takes ~8 hours to do the calibration; in addition, the process has to be repeated if any settings of Wii controllers have been changed.

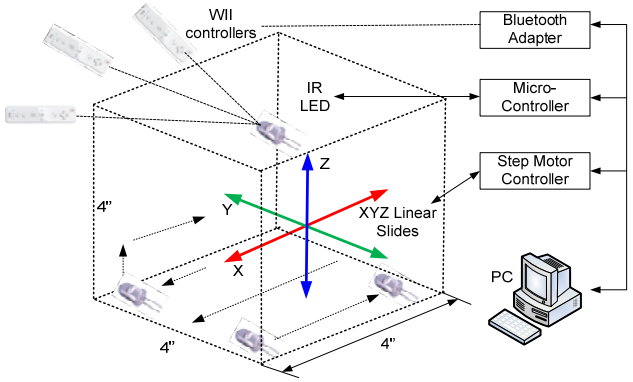


Figure 7: Building database of sampling points by using a 3-axis linear motion device.

Table 2: Example of collected samples: 3D coordinate (X, Y, Z) can be calculated by multiplying the step size (i.e. 5mm). In the right side, 2D coordinates (x, y) are collected from Wii controllers (Wii #1~4).

Coordinate(XYZ)	Real coordinate (mm)	Collected data from Wii (x1,y1,x2,y2,x3,y3,x4,y4)
[1] [7] [5]	5, 35, 25	474,276,288,251,325,332,354,183
[1] [8] [5]	5, 40, 25	474,307,288,275,325,359,355,207
[1] [9] [5]	5, 45, 25	475,338,288,300,326,384,356,232
[1] [10] [5]	5, 50, 25	475,369,287,323,327,412,356,256
[1] [11] [5]	5, 55, 25	475,400,287,348,326,440,352,282
[1] [12] [5]	5, 60, 25	475,433,286,372,326,466,352,307

(2) Computing 3D coordinate by using a regression method

The collected data was examined. The results of 2D points (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) in XY, XZ, and YZ layers in the sampling space show strong linear characteristics. Figure 8 shows the data (x_2, y_2) collected from Wii controller #2 for 20 sampled XZ layers. In the figure, colors that are closer to red mean larger value while colors that are closer to blue mean smaller value.

Based on our data analysis, we concluded that the collected x, y values of Wii controllers are gradually changing between layers. That is, the differences between two neighboring layers are quite small while the differences between layer 1 and 20 are much larger. In addition, x, y values within a same layer are changing gradually and close to linear. Accordingly a two-step fitting approach has been developed to compute a 3D coordinate for given 2D point input.

(1) We first use all the sampling points in the working volume to build a linear regression model to compute an estimate coordinate (X', Y', Z'). Suppose Φ matrix is our collected sample data:

$$\Phi = \begin{bmatrix} 1 & x_1^1 & y_1^1 & x_2^1 & y_2^1 & x_3^1 & y_3^1 & x_4^1 & y_4^1 \\ 1 & x_1^2 & y_1^2 & x_2^2 & y_2^2 & x_3^2 & y_3^2 & x_4^2 & y_4^2 \\ & & & & \cdots & & & & \\ 1 & x_1^{8000} & y_1^{8000} & x_2^{8000} & y_2^{8000} & x_3^{8000} & y_3^{8000} & x_4^{8000} & y_4^{8000} \end{bmatrix}_{8000\times 9}$$

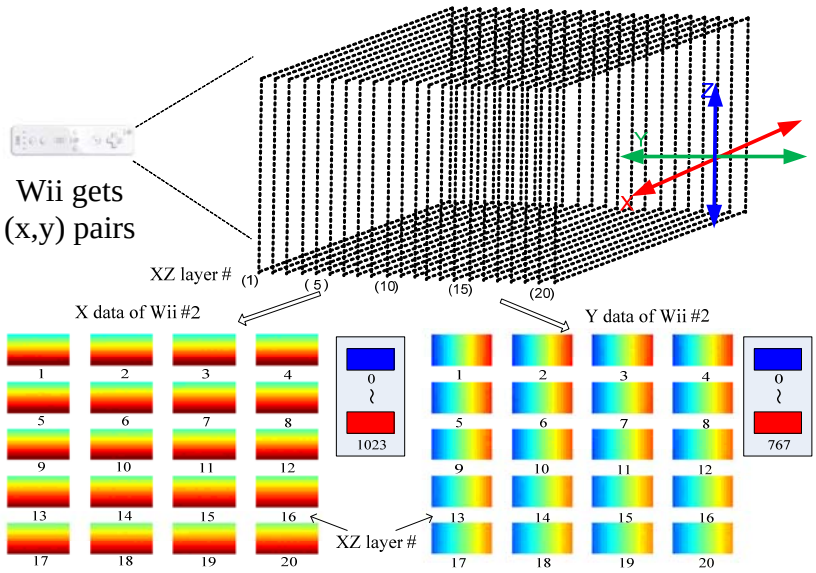


Figure 8: Graphical representation of collected data in *XZ* layers.

And Γ matrix is the related 3D coordinate value of IR-LED in *X*-, *Y*-, or *Z*-axis:

$\Gamma_X = \begin{bmatrix} X^1 \\ X^2 \\ \cdots \\ X^{8000} \end{bmatrix}, \Gamma_Y = \begin{bmatrix} Y^1 \\ Y^2 \\ \cdots \\ Y^{8000} \end{bmatrix}, \Gamma_Z = \begin{bmatrix} Z^1 \\ Z^2 \\ \cdots \\ Z^{8000} \end{bmatrix}$. Hence a set of weight values can be computed as:

$$\beta_X = (\Phi^T\Phi)^{-1}\Phi^T\Gamma_X, \beta_Y = (\Phi^T\Phi)^{-1}\Phi^T\Gamma_Y, \beta_Z = (\Phi^T\Phi)^{-1}\Phi^T\Gamma_Z.$$

The weight values β is a 9×1 vector $\begin{pmatrix} \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & \beta_9 \end{pmatrix}^T$.

For any given 2D point input $\eta = \begin{pmatrix} 1 & x_1 & y_1 & x_2 & y_2 & x_3 & y_3 & x_4 & y_4 \end{pmatrix}$, we know:

$$X' = \eta \times \beta_X, Y' = \eta \times \beta_Y, Z' = \eta \times \beta_Z.$$

(2) The estimated coordinate (*X'*, *Y'*, *Z'*) can give us a good estimation. Accordingly, we can easily find the *XY*, *XZ*, and *YZ* layers that are the closest to the estimated point from *Z*, *Y*, and *X* axis respectively. Hence a better linear regression model can be built based on the points in the identified layers. For example, to compute the *Y* value of the touching point, we can only consider the 400 sampling points in the related *XZ* layer. That is,

$$\Phi = \begin{bmatrix} 1 & x_1^1 & y_1^1 & x_2^1 & y_2^1 & x_3^1 & y_3^1 & x_4^1 & y_4^1 \\ 1 & x_1^2 & y_1^2 & x_2^2 & y_2^2 & x_3^2 & y_3^2 & x_4^2 & y_4^2 \\ & & & & \cdots & & & & \\ 1 & x_1^{400} & y_1^{400} & x_2^{400} & y_2^{400} & x_3^{400} & y_3^{400} & x_4^{400} & y_4^{400} \end{bmatrix}_{400\times 9} \text{ and } \Gamma_Y = \begin{bmatrix} Y^1 \\ Y^2 \\ \cdots \\ Y^{400} \end{bmatrix}.$$

Similarly, $\beta_Y = (\Phi^T\Phi)^{-1}\Phi^T\Gamma_Y$, and $Y = \eta \times \beta_Y$. The *X* and *Z* values can also be computed. Hence a refined coordinate (*X*, *Y*, *Z*) can be returned as the final 3D coordinate of the touching point.

Notice in the above computing processes, we assumed that all 4 Wii controllers have inputs. The approach can easily be extended to other cases that only two or three Wii controllers have inputs. For example, suppose only three Wii controllers have input (suppose no Wii #3). Hence, the given 2D point input $\eta = \begin{pmatrix} 1 & x_1 & y_1 & x_2 & y_2 & x_4 & y_4 \end{pmatrix}$. Accordingly Φ matrix can be modified by deleting x_3 and y_3 columns to generate a matrix whose size is 8000×7 or 400×7. Accordingly the computed weight values β would be a 7×1 vector. Hence the 3D coordinate based on the three Wii controllers is $\begin{pmatrix} X & Y & Z \end{pmatrix} = \begin{pmatrix} \eta \times \beta_X & \eta \times \beta_Y & \eta \times \beta_Z \end{pmatrix}$.

Notice all the above weight values β for different combinations of Wii controllers and different layers can be pre-calculated. All the computed weight values are then indexed and saved in a database. Hence when using the Touch-RE system, the computing of coordinate (*X*, *Y*, *Z*) based on a given 2D point input η is trivially fast, since only matrix multiplications are required.

4.3. Test Result

Based on the presented approach, we computed the estimated coordinate (*X*, *Y*, *Z*) for the 8,000 collected samples and 1,000 verification samples. Since the exact coordinate values are known, our predicted values can be precisely compared with them. We computed the errors based on different usage scenarios for all the 9,000 sampling points. The resulted error statics are shown in Table 3 and 4. In Table 3, the results of using 2 Wii controllers are presented; in Table 4, the results of using 3 or 4 Wii controllers are presented. Based on them, we can see that more input from Wii controllers will give us better accuracy. In addition, the results given by our method are satisfactory. The error of a 3D point input is generally less than 1mm. The cases where average error or standard deviation exceeds 1mm are highlighted in red. We believe better regression models may be able to further improve the accuracy of our system.

Table 3: Error distance (mm) by using two Wii controllers

	Errors (Wii: #1, #2)		Errors (Wii: #2, #3)		Errors (Wii: #3, #4)		Errors (Wii: #1, #3)		Errors (Wii: #1, #4)	
	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std
X	2.1420	1.9675	2.2086	1.8278	2.2618	1.9829	1.9645	1.7439	1.1549	1.0657
Y	0.8220	0.5204	0.4526	0.4982	1.1512	1.4899	0.8273	1.1703	0.9387	0.9040
Z	0.2896	0.4699	0.2896	0.4699	1.4071	2.032	0.7289	1.0109	0.9626	1.3385

Table 4: Error distance (mm) by using three and four Wii controllers

	Errors (Wii: #1, #2, #3)		Errors (Wii: #1, #2, #4)		Errors (Wii: #1, #3, #4)		Errors (Wii: #2, #3, #4)		Errors (Wii: #1, #2, #3, #4)	
	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std
X	1.3471	1.1428	0.2202	0.2409	0.2511	0.2529	1.6780	1.4173	0.2176	0.2347
Y	0.7940	0.1061	0.8050	0.4778	0.2788	0.1398	0.3398	0.2835	0.2271	0.1011
Z	0.2975	0.2010	0.3995	0.3293	0.2798	0.2577	0.3021	0.1621	0.2505	0.2554

In addition to the error statistic values, the histogram of error distribution for the case of input with 4 Wii controllers is also shown in Figure 9. It can be seen that, for the 9000 samples in the working volume, most errors are within 0 ~ 0.25mm range. Based on the test results, it can also be seen that, for the best accuracy, the user needs to rotate the object such that a touching point can be sensed by all the four Wii controllers.

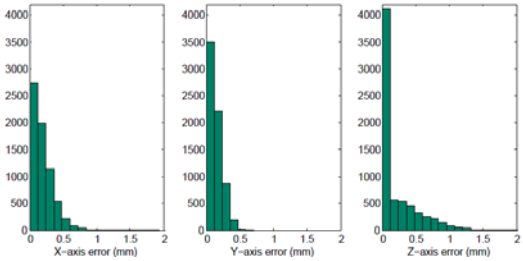


Figure 9: Histogram of error distribution

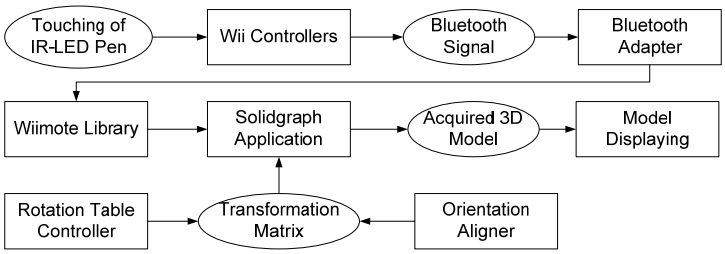


Figure 10: Data flow in our software system.

5. SOFTWARE OF TOUCH-RE SYSTEM

Several software libraries and drivers have been integrated in the Touch-RE system. Figure 10 illustrates the data flow in our system. Starting from a IR-LED pen, Wii controllers dynamically sent Bluetooth signal on its current position. When a personal computer (PC) gets the Bluetooth signal from a Bluetooth adapter, the signal is decoded into readable information. Then 3D point coordinate of the touching point is computed from the input of multiple Wii controllers. Accordingly our application can create 3D models

based on the touching input, which are dynamically displayed to the user. Two other modules, rotation table controller and orientation aligner (refer to Section 6.2), can interact with the application by updating the transformation matrix of the 3D models.

5.1. Wiimote Library

We used BlueSoleil software (www.bluesoleil.com) to receive Bluetooth radio signal sent by Wii controllers. To decode machinery signal from Wii controllers, we used a C++ base Wiimote library called 'Wiiyourself' [15]. The library converts device's signal into a pre-defined C++ class data structure. We also adopted WDK (Windows Driver Kit) for supporting Wiiyourself library. Hence our application can easily get x and y coordinate of each Wii controller from the library. The library's capability on supporting multiple Wii controllers was tested. Based on our experiment, it was found that the library can support and accept signal from 4 Wii controllers simultaneously without any problem.

5.2. Open-Source CAD application: Solidgraph

We selected Solidgraph CAD software as the shape modeling tool in our system. Developed by *Geometros geometrical systems*, Solidgraph is an open-source system for 3D modeling of complex geometric objects [16]. The application supports building of primitives such as point, line, circle, and spline. It also supports Booleans and other Kinematic operations. The open-source application is available from: http://www.codeproject.com/KB/applications/solidgraph.aspx. To work with our 3D point input by IR-LED pens, we modified the drawing functions provided by the system such that models can be dynamically updated based on touching point input. For example, a 'touch to model a box' function is added, which requires users to indicate two corner points of the box (e.g. bottom left and top right). Hence touching information can be seamlessly integrated in our system.

5.3. Rotation Table Controller

An individual application, rotation table controller, has been developed based on Phidget libraries in C. The controller has an intuitive interface to allow users to freely rotate the real object in both directions (clockwise and counterclockwise). Accordingly, the desired motor movement is converted into a set of commands defined by Phidget's Application Program Interface (API) library. The motor movement commands are then sent via a USB cable and executed by the servo-motor controller board. At the same time, based on the recorded rotation angle, the transformation matrix of the coordinate system can be dynamically updated. The rotation table controller can also receive rotation request from the Solidgraph application. Hence the rotation table will rotate when the virtual model in Solidgraph application has been rotated by the user in its graphical user interface (GUI).

6. 3D MODEL GENERATION BY TOUCH-RE SYSTEM

Based on the presented hardware and software systems, we discuss the use of Touch-RE in generating 3D models from touching input. Model alignment is mainly discussed for synchronizing the virtual model with the real object whose positions may be modified.

6.1. 3D Shape Definition in Touch-RE

We implemented five 2D primitives and eight 3D primitives in Solidgraph application. Table 5 shows their definition and possible scenario of creating models by using touch input. As shown in the table, almost all the primitives can be constructed by no more than three touches of object. 'Location of points' means the minimum information to build models and 'Optional point' means more accurate shape can be built by providing more points. For example, for defining a circle, our application can accept just two points at a minimum (i.e. left and right points of the circle). However, a user can also touch one more point on the circle if the exact left and right point of circle cannot be positioned. Accordingly our application can fit circle shape based on the inputs.

Table 5: Definition of primitives and their usage scenario based on touching inputs.

Primitive name	# of points	Dimension	Location of points	Optional point
Point	1	2D	Start	-
Line	2	2D	Start, end	-
Circle	2	2D	Left, Right	Yes
Arc	3	2D	Start, end, radius	-
Spline	Many	2D	Points as much user wants	-
Box	2	3D	Bottom left, top right	-
Sphere	2	3D	Middle left, middle right	Yes
Cylinder	3	3D	Bottom left, bottom right, top left	Yes
Cone	4	3D	Bottom left, bottom right, top left, top right	Yes
Torus	3	3D	Center, outside point, inside point	Yes
Ellipsoid	5	3D	Left, right, top, bottom, height	Yes
Spherical band	4	3D	Middle left, middle right, top, down	Yes
Extrude Op.	1	2D, 3D	End	-

6.2. Alignment between Various Orientations

As discussed in Section 2, real and virtual objects co-located in our system. Their correlated positions are known to our system. Even limited motion has been provided by the rotation table, the real object may still need to be flipped or re-oriented by the user. For example, if an object needs to be flipped 90 degrees from position 1 to 2 in order to access a face which is in the bottom side in position 1. Obviously the established position alignment between the virtual and real objects has been broken by the flipping. Their correlated positions need to be identified such that the virtual and real objects can be re-aligned.

The basic idea in the position alignment is to identify markers that are known to both virtual and real objects such that a transformation matrix can be computed based on them. The markers should be easy to identify and can be precisely touched by the user. Based on the types of such markers, two alignment techniques have been developed in our system.

(1) Corner method

The most easily identified and touched markers are corners. Hence the first method uses the information of 4 or more points from users. An example is shown in Figure 11. Our goal is to compute a homogeneous matrix H (4×4) which consist of translation and rotation information such that:

$$\text{Original object} \times H = \text{Moved object} \tag{6.1}$$

Assume the real object is rigid. Hence the same transformation can be applied to all the points in the model. For example, if user indicated 4 points (O_1, O_2, O_3, O_4) in the original virtual model and related points (N_1, N_2, N_3, N_4) in the re-positioned real object, we know:

$$H = \begin{bmatrix} N_{x1} & N_{x2} & N_{x3} & N_{x4} \\ N_{y1} & N_{y2} & N_{y3} & N_{y4} \\ N_{z1} & N_{z2} & N_{z3} & N_{z4} \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} O_{x1} & O_{x2} & O_{x3} & O_{x4} \\ O_{y1} & O_{y2} & O_{y3} & O_{y4} \\ O_{z1} & O_{z2} & O_{z3} & O_{z4} \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \tag{6.2}$$

Hence the virtual model can be realigned by applying the homogeneous matrix H. After 4 points are indicated, the user can see the realigned object on screen. Notice, however, there may be slight errors in the 3D coordinates of (N_1, N_2, N_3, N_4) since they are input by touching on the re-positioned real object. This will lead to errors in the alignment. To address the problem, we also allow users to input more than 4 points in our system. For example, if user indicated 6 points (O_1, O_2, O_3, O_4, O_5, O_6) in the original virtual model and related points (N_1, N_2, N_3, N_4, N_5, N_6) in the re-positioned real object, we can use the pseudo-inverse method to compute:

$$H = \begin{bmatrix} N_{x1} & N_{x2} & N_{x3} & N_{x4} & N_{x5} & N_{x6} \\ N_{y1} & N_{y2} & N_{y3} & N_{y4} & N_{y5} & N_{y6} \\ N_{z1} & N_{z2} & N_{z3} & N_{z4} & N_{z5} & N_{z6} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} O_{x1} & O_{x2} & O_{x3} & O_{x4} & O_{x5} & O_{x6} \\ O_{y1} & O_{y2} & O_{y3} & O_{y4} & O_{y5} & O_{y6} \\ O_{z1} & O_{z2} & O_{z3} & O_{z4} & O_{z5} & O_{z6} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \tag{6.3}$$

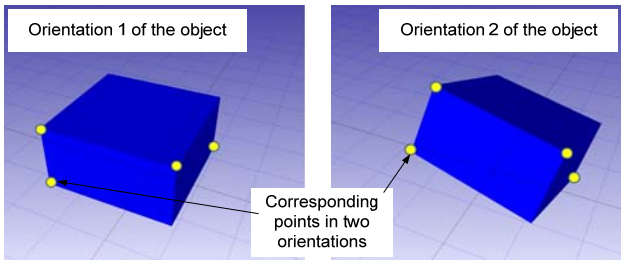


Figure 11: Example of vector method.

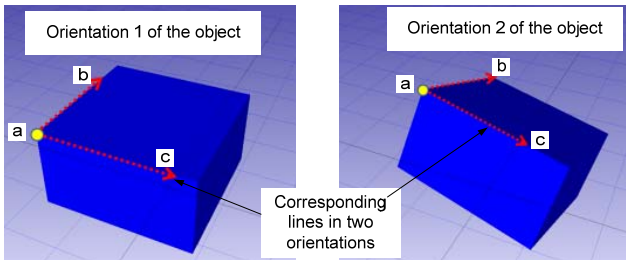


Figure 12: Example of vector alignment.

(2) Vector method

Another type of markers that can be easily identified and touched is edges. Vector method is based on two vectors related to two straight edges in the model. As shown in Figure 12, if we know the positions of two vectors (*ab*, *ac*) in the original virtual model and the re-positioned real object, a homogeneous matrix *H* can be computed. In many cases, we found it is much faster to define two vectors than to define multiple points. One main reason is that it usually requires less rotation and interaction with the PC when defining two vectors compared with defining 4 or more points.

7. TEST EXAMPLES

In the section, we present two examples to demonstrate the capability of our testbed system. Both examples are based on common objects taken from our personal life. Hence they are applicable to *personal manufacturing*. In the first example, we used an eye drop bottle which is relative simple to illustrate the idea. In the second example, we used a toy car with multiple components. Its modeling process requries alignments between different orientatons for the same object, and also positioning between various objects.

7.1. Test 1: Building an Eye Drop Bottle

An eye drop bottle consists two cylinders and one cone. The model can be built without table rotation. The building process is shown in Figure 13. In Figure 13.(a), we can construct a cylinder by touching 3 points; in Figure 13.(b), another cylinder is constructed in a similar way; finally, a cone with a smaller size is constructed by touching 4 points. Notice the relative positon of the three features are determined by the touching points. The result indicates that our system can accuately capture the 3D coordinates of the touching points defined by the IR-LED pen.

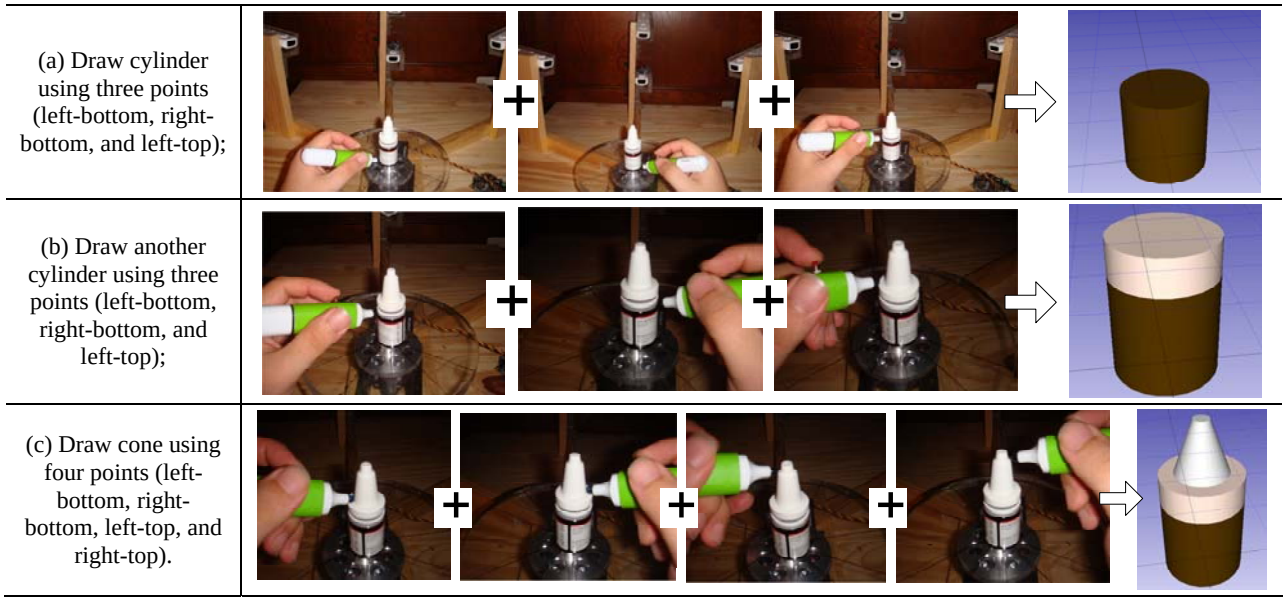


Figure 13: Model acquiring process of an eye drop bottle with multiple features.

7.2. Test 2: Building a Toy Car

The 3D model of a toy car with multiple components is reconstructed in this example. The model acquiring process requires us to use different primitives, Boolean and extrusion operations, rotating table, and alignment functions. The building process is shown in Figure 14. We first constructed the CAD model of each component; based on touching input, we then align them into one assembled model.

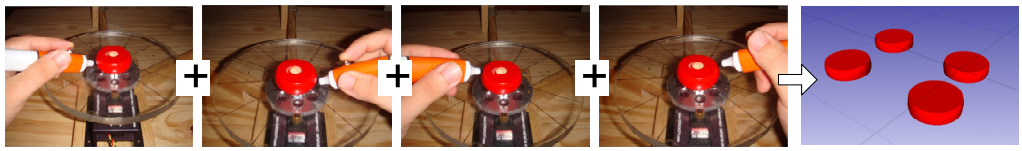
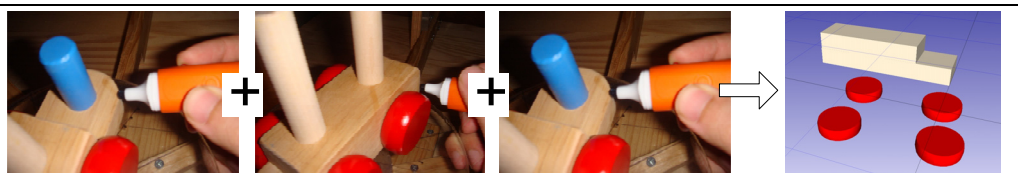
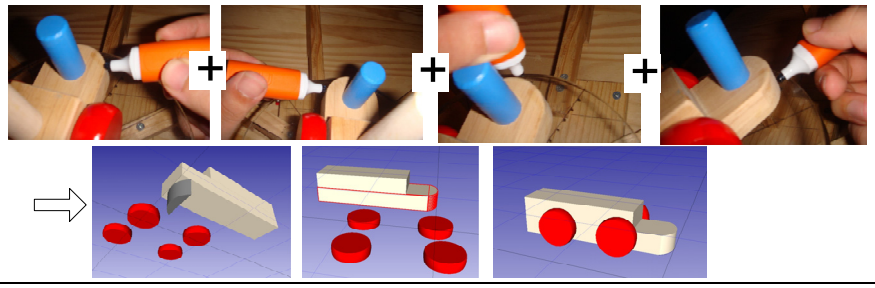
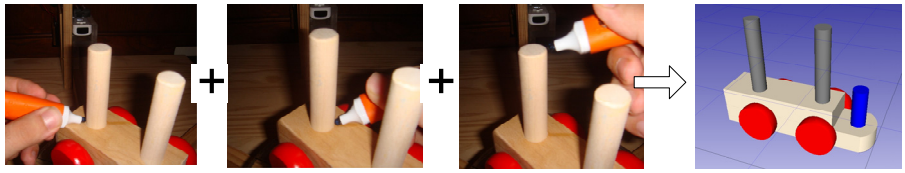
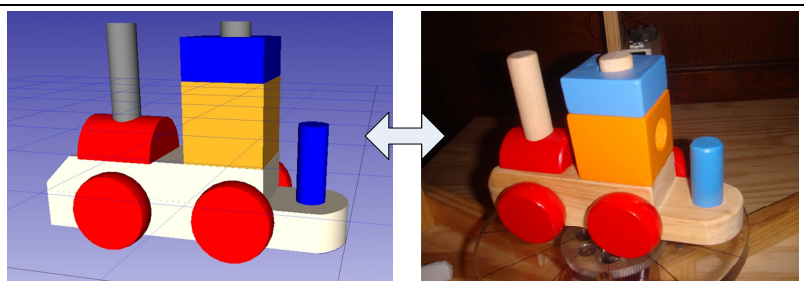
(a) Draw wheels (sphereical band) using four points (left-bottom, right-bottom, left-top, and right top);	
(b) Draw two boxes using three points (left-bottom, right-bottom, and left-top);	
(c) Draw round shape using contour and extrude operation. After this, we combined new shape with box using Boolean operations. Then put the wheel using alignment function;	
(d) Draw three cylinders using three points (left-bottom, right-bottom, and left-top);	
e) Draw half-cylinder using contour and extrude operation;	
f) Generate two more boxes and put the half-cylinder to the right position. Finally the acquired models (left) and the real objects (right).	

Figure 14: Model acquiring process of a toy with multiple components.

8. CONCLUSION

In this paper we have presented a reverse engineering system that is based on touching input from users. The system is extremely low-cost that uses commercial devices (Wii controllers) and IR-LED pens to obtain 3D touching points. Based on our system, users can easily interact with both real and virtual objects and quickly acquiring feature-based CAD models. To achieve a desired accuracy, we present a regression based 3D coordinate computation method based on the input of an arbitrary number of Wii

controllers. The test results indicate that the error of a 3D touching point in our system is generally less than 1mm. We also present various hardware and software approaches for improving users' capability on defining touching input such as the integration of a rotation table and different object alignment algorithms. Two examples were given to illustrate the effective of our system.

One limitation of our current system is that is has limited capability on processing complex freeform features. Currently we mainly consider typical mechanical features and simple freeform surface that can be approximated by B-spline surfaces. We are exploring various approaches including integrating a 3D scanner in our system to expand its capability.

ACKNOWLEDGEMENT

The authors would like to thank *Jingyuan Lao*, a master student in our lab, for his help on building the hardware system.

REFERENCES

1. MGX Lamps: http://www.materialise.com/MGX.
2. CafePress: http://mugs.cafepress.com/.
3. Shapeways – Passionate about creating: http://mugs.shapeways.com/.
4. Spore Sculptor: http://www.sporesculptor.com/.
5. Varady, T., R. Martin, and J. Cox. Reverse Engineering of Geometric Models - an Introduction. Computer-aided Design, Vol. 29, No. 4, pp. 255–268, 1997.
6. Milgram, P. and F. Kishino. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems*, E77-D(12), pp.1321-1329, 1994.
7. Lu, S. C. Y., M. Shpitalni, and R. Gadh. Virtual and Augmented Reality Technologies for Product Realization. CIRP Annals – Manufacturing Technology, Vol. 48, No. 2, pp. 471–495, 1999.
8. Azuma, R., Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent Advances in Augmented Reality. Computer & Graphics, Nov. 2001.
9. Bimber, O., and R. Raskar. Spatial Augmented Reality: Merging Real Virtual Worlds. A K Peters Ltd., Massachusetts, MA, USA, 2004.
10. Vallino, J. R. Interactive Augmented Reality. PhD dissertation, Department of Computer Science, University of Rochester, Rochester, NY, 1998.
11. Regenbrecht, H., C. Ott, M. Wagner, T. Lum, P. Kohler, W. Wilke, E. Mueller. An Augmented Virtuality Approach to 3D Videoconferencing. In *Proceedings of the 2nd IEEE/ACM international Symposium on Mixed and Augmented Reality*. IEEE Computer Society, pp. 290-291, 2003.
12. Lee, J. C. Hacking the Nintendo Wii Remote. IEEE Pervasive Computing, 7(3):39–45, 2008.
13. Hay, S., Newman, J., and Harle, R. Optical tracking using commodity hardware. In *Proceedings of the 7th IEEE/ACM international Symposium on Mixed and Augmented Reality*. Symposium on Mixed and Augmented Reality. IEEE Computer Society, Washington, DC, 159-160, 2008.
14. Zhang, Z. Flexible camera calibration by viewing a plane from unknown orientations. International Conference on Computer Vision (ICCV'99), Corfu, Greece, pages 666–673, 1999.
15. http://wiiyourself.gl.tter.org/.
16. http://www.geometros.com/solidgraph/index.htm.