symmetry

MDPI

Article

Tangible Visualization Table for Intuitive Data Display

Jongyong Kim, Cheongun Lee, Seung-Hyun Yoon and Sanghun Park *

Department of Multimedia, Dongguk University, 30, Pindong-ro 1gil, Jung-gu, Seoul 04620, Korea; khj8499@dongguk.edu (J.K.); delphi1004@dgu.ac.kr (C.L.); shyun@dongguk.edu (S.-H.Y.)
* Correspondence: mshpark@dongguk.edu; Tel.: +82-2-2260-3765

Received: 18 October 2017; Accepted: 9 December 2017; Published: 13 December 2017

Abstract: We propose a new tangible visualization table for intuitive and effective visualization of terrain data transferred from a remote server in real time. The shape display approximating the height field of remote terrain data is generated by linear actuators, and the corresponding texture image is projected onto the shape display. To minimize projection distortions, we present a sophisticated technique for projection mapping. Gesture-based user interfaces facilitate intuitive manipulations of visualization results. We demonstrate the effectiveness of our system by displaying and manipulating various terrain data using gesture-based interfaces.

Keywords: tangible system; visualization; shape display; actuator; projection mapping; gesture interface

1. Introduction

Over the past several decades, visualization technologies and display devices have progressed rapidly. Many people have become familiar with display devices such as high-resolution 3D TVs and HMDs (head-mounted displays). Recently, Microsoft has developed Hololens for shared holographic experiences, which realizes AR (augmented reality) without shielding the user from the real world. While these technologies provide the users with realistic and immersive experiences, they might cause side effects such as headache, dizziness and eye fatigue. Moreover, their usages are somewhat limited to general VR and AR contents, and natural user interactions are not easy to implement. Therefore, there is still need for the development of a specialized visualization system to meet the user's specific purpose.

Intuitive and realistic visualization facilitates the interpretation of raw data collected from various experiments, measurements and surveys. Depending on the type of raw data, different techniques [1–11] have been developed, and some of them proposed specialized display devices [3,5,7,9] for enhancing visualization effects. However, these techniques and systems are restricted to visualizing the small-scale data stored in local client devices and do not support the visualization of large-scale data transferred from a remote server in real time.

In this paper, we propose a new tangible visualization table for intuitive and realistic display of remote terrain data using gesture-based user interactions. Figure 1 shows our tangible visualization table. The upper part of the table is composed of a projector and a Kinect for projecting images and recognizing the user's gesture, respectively. The lower part of the table consists of an array of actuators forming a display surface and an array of LEDs (light emitting diodes) to enhance visualization effects.

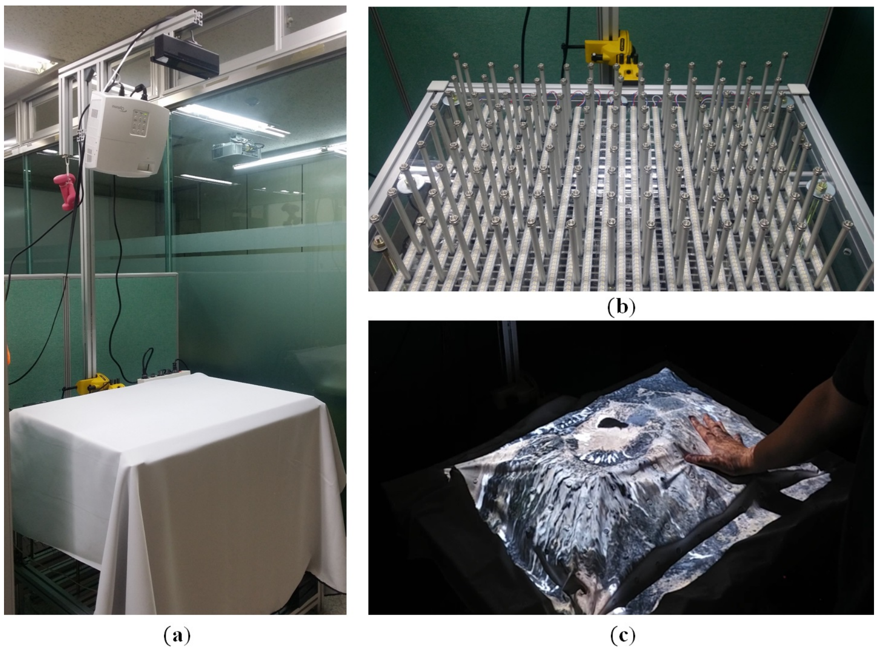


Figure 1. Tangible visualization table: (**a**) a projector and a Kinect on the upper part; (**b**) actuator array on the lower part; (**c**) tangible visualization of terrain data.

We employ the Arduino platform [12] for controlling the linear actuators and LEDs since it enables users to easily implement the required functions by using Arduino IDE (Integrated Development Environment). A display surface similar in shape to the remote terrain data is formed, and the texture image of the terrain is projected onto the display surface while minimizing projection distortion. Operational software for the tangible visualization table has been developed using Unity3d [13] and OpenCV [14], a game engine and an open source library for computer vision and image processing. The user's various gestural inputs are recognized by the Kinect installed on the upper part of the table and are used to effectively observe and manipulate the remote terrain data.

The main contributions of our system can be summarized as follows:

- We develop a new tangible visualization table that supports intuitive and realistic visualization of terrain data transferred from a remote server in real time.
- We propose a sophisticated technique for projection mapping while minimizing mapping distortion of the texture image onto a display surface generated by linear actuators.
- Our system provides an intuitive and efficient control mechanism for visualization of remote terrain data using gesture-based user interfaces.

There exist numerous possible applications of tangible visualization system. One obvious application is geography education. An instructor can obtain the terrain data of an unknown area from a remote server in real time. Then, our system will project the corresponding texture image onto the display surface formed by linear actuators. In this scenario, learners can better understand the geography of an unknown region while touching the display surface. Another possibility is to combine our system with an AR device. Our system can transmit the simulation results of natural disasters such as flood and heavy rains to the AR device while sharing terrain geometries. The users are then able to see the simulation results augmented on the display surface through AR devices.

The structure of this paper is as follows. Section 2 introduces the existing tangible visualization systems. Section 3 describes the hardware structure of the tangible visualization table proposed in this paper and the software modules for operating the system. Section 4 describes in detail the process of visualizing big terrain data in the tangible visualization table, while Section 5 describes the gesture-based user interface technology developed for our system. Experimental results are demonstrated in Section 6 and finally we conclude this paper and propose directions for future research in Section 7.

2. Related Work

The Relief [5,15] system was developed in MIT's Tangible Media Lab. Relief is a 450 mm × 450 mm curved surface display, with 120 actuators arranged on a circular table in a 12 × 12 grid form; the system displays images from the projector by covering an array of the actuators with a screen. Relief supports a camera-based gesture interface as the user-to-system interface. It controls projected images using gestures such as hand clenching and spreading, and allows transferring, zooming in/out and rotating the data represented by the actuators. However, because it projects images in a front projection mode, images from the projector are occluded during the interaction with hands.

Lumen [7] is a system in which 169 actuators are arranged in a 13 × 13 grid form. Lumen does not require an output device like a projector, because it installs an element in the actuators that can represent RGB colors. To allow user interaction, a sensor capable of sensing the touch of a hand is installed directly into the actuators. Lumen has the advantage of being able to communicate between systems, but has its own limitations; the range of information that can be represented is limited by the small display and the actuator height.

Piper et al. [16] proposed an effective system for the real-time computational analysis of landscape models. As the users alter the topography of a clay landscape, its shape is captured by a laser scanner in real time while analyzing it and projecting the result. They showed that this system can effectively be used for landscape design and engineering. The AR sandbox [9] allows the user to create a display by shaping sand. The projector installed in the upper part projects images onto the sand and then uses the Kinect to recognize the shape of the sand in real time; it then shows contour lines and colors according to the height and shape of the sand, as adjusted by the user with his/her hands or tools. In addition, a water flow simulation is run at the position where user gestures are recognized, and the simulation result is influenced in real time by the sand that the user shapes.

The multi-touch display system in [3] is 870 mm × 570 mm × 182 mm in size and consists of a mountain-like shape display. This system displays images from a projector installed on the lower part in a rear projection mode. It prevents display occlusion arising from front projection by adopting rear projection. In addition, the user's position information can be obtained by installing an infrared camera on the lower part where the user touches the display surface. A disadvantage is that the display screen cannot be made malleable, because the screen of this system is fixed.

LEDs are installed in the tangible visualization table system proposed in this paper to address the weaknesses mentioned above; i.e., the display occlusion of the Relief system and the drawback of Lumen in that it cannot display very much information. Images can be displayed both in a front projection mode via a projector and in a rear projection mode via LEDs, so that, compared to existing methods, various data can be effectively visualized. This also minimizes the distortion of images caused by the calibration of the projector and the camera, allowing accurate mapping of images onto the display in real time.

Natural user interfaces (NUIs) play an important role in visualization system. Many sophisticated methods for gesture recognition have been developed in the last decade. Hilliges et al. [17] developed a tabletop system based on a depth camera and holoscreen in which a novel shadow-based user interface is employed for providing feedback during mid-air interactions. However, their system shows unstable results for manipulating virtual objects. Recently, Follmer et al. [18] proposed a method for utilizing shape displays to mediate user interaction. They demonstrated how dynamic affordances, constraints and object actuation can create novel interactions. In this paper, we employ simple, but robust gesture-based interfaces for manipulating the tangible visualization system by using the Kinect camera. Most of users' interactions in our system are restricted to zooming in/out, translating and rotating of terrain data to find a specific region and to see its details. Therefore, our gesture-based user interfaces are sufficient to support the necessary interactions.

3. System Architecture

Figure 2 shows the system configuration of the tangible visualization table system proposed in this paper. The user receives the height field data stored in the remote server either directly via TCP/IP communication or from other visualization clients. After processing them, the tangible visualization table transfers the received data to Arduino and the projector. Created images are visualized via actuators, LEDs and the projector. The Kinect is used for gesture-based user interaction.

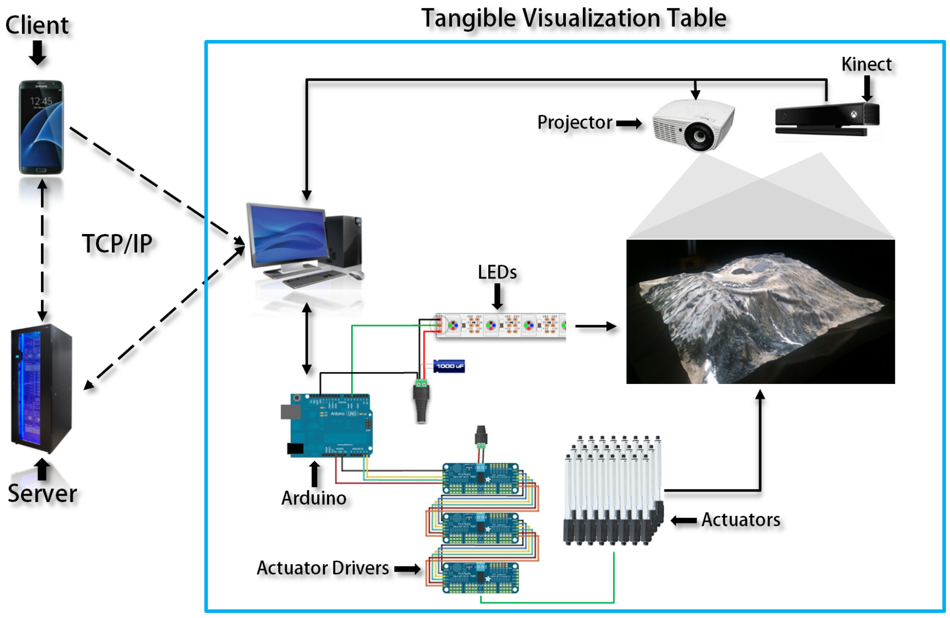
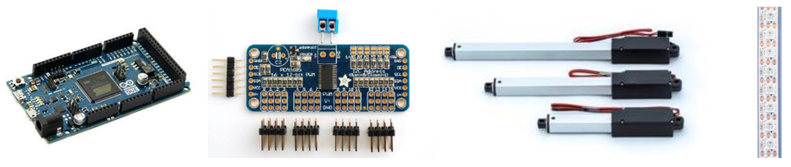


Figure 2. Structure of the tangible visualization system.

3.1. Hardware Structure

The lower part of the system is equipped with various devices such as LEDs, actuators and the Arduino platform, on an 800 mm × 600 mm × 750 mm table. Figure 3a shows Arduino Due, which combines microprocessor and input/output modules. Figure 3b shows the actuator drivers to which the actuators shown in Figure 3c are connected; these are also connected to Arduino Due to allow communication. Up to 16 actuators can be connected to one driver, and up to 62 drivers can be connected to each other; thus, up to 992 actuators can be controlled. Figure 3c shows the linear actuators used in this system. They create up and down motion in a straight line and present the data received from the server on a curved display. The LED device shown in Figure 3d has a length of 1000 mm, with 144 LEDs connected in strip form. This device can control R (red), G (green), B (blue) and W (white) colors.



(**a**) (**b**) (**c**) (**d**)

Figure 3. Hardware devices in the lower part of the system: (**a**) Arduino Due; (**b**) actuator driver; (**c**) actuator and (**d**) LED.

In the upper part of the tangible visualization table, there is a projector to project images and a Kinect to capture images of the user gestures, as well as for calibration of the projector. Figure 4a shows an Optoma EH415ST projector installed on the upper part, which is equipped with a short-focus lens to project a 90-inch screen at a distance of one meter. Figure 4b shows a Kinect V2 model installed on the upper part, which is composed of an RGB camera, IR emitter and IR depth sensor. If the IR emitter emits infrared rays, the IR depth sensor detects infrared rays reflected by an object, and the three-dimensional depth is calculated. In this system, the Kinect is used to recognize the user's gestures and to analyze the shape of the generated surface of the display.

(**a**) (**b**)

Figure 4. Hardware devices in the upper part of the system: (**a**) projector and (**b**) Kinect V2.

3.2. System Software

Two types of system software are required to operate the tangible visualization table. One, created by Unity3d, runs on a PC, while the other is Arduino-based software that controls actuators and LEDs while communicating with a PC.

Figure 5 shows the overall system software structure. The Kinect input data are recognized as user gestures by the Image processor module and the Hand info module [19]. The system either executes the corresponding menus or selects an area of interest through operations such as transferring or zooming in/out based on the recognized gestures and then receives data from the remote server. The received data are sampled by the Geometry generator module and transferred to the Arduino software by the Sender module.

Protocols are required for data communication between the PC and the Arduino software. Table 1 shows the protocols for data communication between them; each data item is separated by a blank space and transferred to the Arduino. Figure 6 shows a flowchart of the Arduino program that sends and receives data to and from the PC using serial communication. The Arduino software analyzes the received data according to the protocols and runs the actuators and LEDs based on the analyzed results.

Table 1. Protocol contents.

Protocol Name	Description
System ID	Division instructions for LED and actuator
Command ID	Promised instructions between Arduino and PC
State	State of Arduino and PC
Data Length	Length of transferred data
Data	Actual transferred data

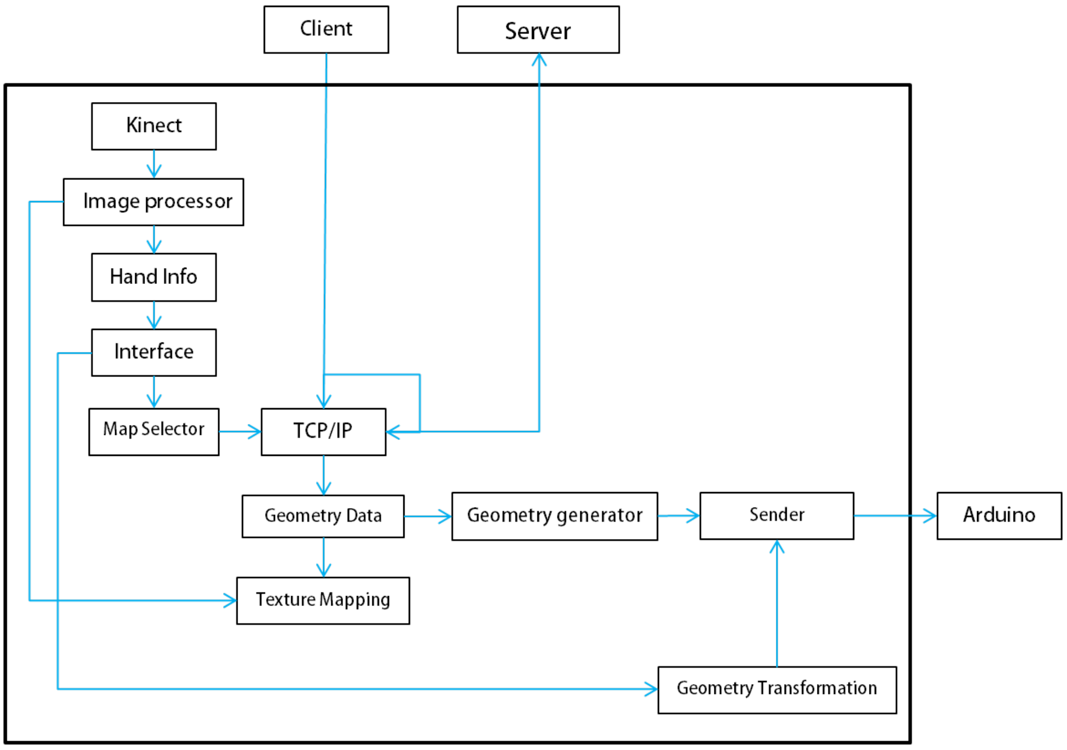


Figure 5. Software structure of the tangible visualization table.

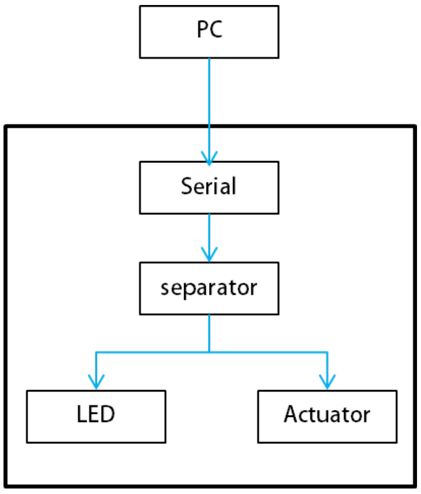


Figure 6. Internal structure of Arduino.

4. Projection Mapping

Projection mapping is a technique for displaying a 2D image on a non-flat surface and has been widely used for many applications such as advertisement, live concerts and theater [20]. Our tangible visualization table projects terrain texture images received from the data server onto a curved surface generated by moving actuators. Therefore, we need a sophisticated projection technique for minimizing the mapping distortion.

4.1. Calibration of the Kinect with the Projector

The Kinect and projector use different coordinate systems, and thus, the calibration between them should be performed when projecting a texture image onto a curved display generated by actuators [21,22]. Let $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ be the homogeneous coordinates of $\mathbf{p}$ measured by the Kinect depth (infrared) camera and $\begin{bmatrix} wx & wy & w \end{bmatrix}^T$ be the perspective coordinates of the projection point $\mathbf{p}'$. The transformation between $\mathbf{p}$ and $\mathbf{p}'$ can be represented as follows:

$$\mathbf{p}' = A\begin{bmatrix} R & | & \vec{t} \end{bmatrix}\mathbf{p} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1}$$

where $A \in \mathbb{R}^{3\times3}$ denotes the intrinsic parameters and $R \in \mathbb{R}^{3\times3}$ and $\vec{t} \in \mathbb{R}^3$ denote the extrinsic parameters, respectively. Once these transformation matrices have been obtained, the three-dimensional coordinates measured by the Kinect depth camera can be converted to the two-dimensional coordinates of the projector.

We use the GML C++ Camera Calibration Toolbox [23] to find the intrinsic parameters of the Kinect. They can be obtained using images taken at various positions and angles of paper on which a chess board is printed. Radial distortion and tangential distortion [24] of an image can be reduced using the calculated intrinsic parameters. Table 2 lists the intrinsic parameters of the Kinect obtained through this method.

Table 2. Calculated intrinsic parameter values.

Intrinsic Parameters	Calculated Values
(f_x, f_y)	(1058.905250, 1059.267839)
(c_x, c_y)	(959.382450, 538.959259)
Distortion coefficient	(0.014551, −0.003946)

To find the extrinsic parameters, we employ the matching points between the 3D real-world coordinate system and the image coordinate system of the projector. The matching points are found by using each corner point in a commonly-used chessboard pattern. The OpenCV function is used to find the corner points of the chess board [14]. The Kinect infrared camera is required to find the depth value of the real-world coordinate system. Because the infrared camera cannot read RGB color values, the Kinect RGB camera is used to locate corner points; the points of the 3D real-world coordinate system are then found by converting the located points to the coordinates of the Kinect infrared camera.

Figure 7 shows the mapping of the points found in the RGB camera to the infrared camera. To match the points of the real-world coordinate system to the image coordinate system of the projector, the projector must know the corner points of the projected chessboard and project the fixed chessboard image. A plate onto which images are projected is installed to find the points of the real-world coordinate system, and the corner points are then found while moving and rotating the plate. With N matching points obtained in this way, the calibration is processed.

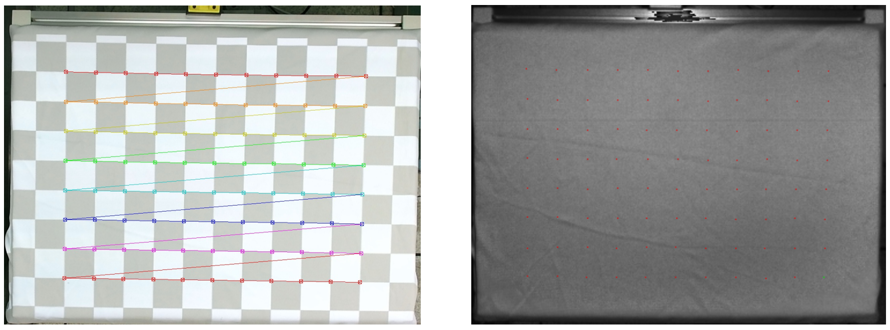

(**a**) (**b**)

Figure 7. Coordinate mapping: (**a**) corner points of the RGB camera; (**b**) corner points of the infrared camera.

Equation (2) represents the relationship when three-dimensional coordinates $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ are projected onto two-dimensional coordinates $\begin{bmatrix} x_p & y_p & 1 \end{bmatrix}^T$. The extrinsic parameters are represented by a 3 × 3 rotation matrix and a 3 × 1 translation matrix.

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

Equation (2) can be expressed as Equation (3), where coefficients (c_1, c_2) are used to match the Kinect infrared camera resolution ratio and the projector resolution ratio.

$$\begin{aligned} x_p &= \frac{c_1 r_{11} X + c_1 r_{12} Y + c_1 r_{13} Z + c_1 t_1}{r_{31} X + r_{32} Y + r_{33} Z + t_3}, \\ y_p &= \frac{c_2 r_{21} X + c_2 r_{22} Y + c_2 r_{23} Z + c_2 t_2}{r_{31} X + r_{32} Y + r_{33} Z + t_3}. \end{aligned} \tag{3}$$

Then, Equation (4) can be obtained by dividing the numerator and denominator by t_3.

$$\begin{aligned} x_p &= \frac{q_1 X + q_2 Y + q_3 Z + q_4}{q_9 X + q_{10} Y + q_{11} Z + 1}, \\ y_p &= \frac{q_5 X + q_6 Y + q_7 Z + q_8}{q_9 X + q_{10} Y + q_{11} Z + 1}, \end{aligned} \tag{4}$$

where $q_1 = \frac{c_1 r_{11}}{t_3}, q_2 = \frac{c_1 r_{12}}{t_3}, \ldots, q_{11} = \frac{r_{33}}{t_3}$.

Equation (5) below can be obtained by multiplying both sides of Equation (4) by the denominator.

$$\begin{aligned} q_9 X x_p + q_{10} Y x_p + q_{11} Z x_p + x_p &= q_1 X + q_2 Y + q_3 Z + q_4, \\ q_9 X y_p + q_{10} Y y_p + q_{11} Z y_p + y_p &= q_5 X + q_6 Y + q_7 Z + q_8. \end{aligned} \tag{5}$$

If we leave only x_p and y_p on the left side, then Equation (5) can be expressed as:

$$\begin{aligned} x_p &= q_1 X + q_2 Y + q_3 Z + q_4 - q_9 X x_p - q_{10} Y x_p - q_{11} Z x_p, \\ y_p &= q_5 X + q_6 Y + q_7 Z + q_8 - q_9 X y_p - q_{10} Y y_p - q_{11} Z y_p. \end{aligned} \tag{6}$$

If we plug in all coordinates of matching points into Equation (6), we obtain a system of linear equations expressed as follows:

$$\begin{bmatrix} x_{p1} \\ y_{p1} \\ x_{p2} \\ \vdots \\ y_{pn} \end{bmatrix} = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_{p1} & -Y_1x_{p1} & -Z_1x_{p1} \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_{p1} & -Y_1y_{p1} & -Z_1y_{p1} \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -X_2x_{p2} & -Y_2x_{p2} & -Z_2x_{p2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_{pn} & -Y_ny_{pn} & -Z_ny_{pn} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{11} \end{bmatrix}. \tag{7}$$

Since the number of matching points is greater than 11, the above linear system is over-determined, and thus, the unknowns $q_i, i = 1, 2, \ldots, 11$ can be found by using the QR decomposition method [25]. Finally, the extrinsic parameters of the Kinect can be determined from q_i as follows:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} = \begin{bmatrix} -3.066175 & -0.108339 & 0.001503 & 1877.717770 \\ 0.009460 & -3.157402 & 0.611820 & 1040.382690 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

4.2. Texture Deformation

Texture deformation is needed to accurately project texture images onto a curved surface made by actuators. When projecting the raw texture image received from the server onto a curved display, the color is not projected at the correct position. Figure 8 compares the results of projecting the texture image using different methods. Figure 8a shows the projection of the raw texture image without deformation, while Figure 8b shows the result of projecting the deformed texture using the extrinsic parameters obtained in Equation (8). The red circle in Figure 8 represents the position of an actuator; if the texture is projected at the correct position, the image must be projected onto the actuator. Figure 8b shows the result of precise image projection on the actuator after deforming texture using the extrinsic parameters in Equation (8).

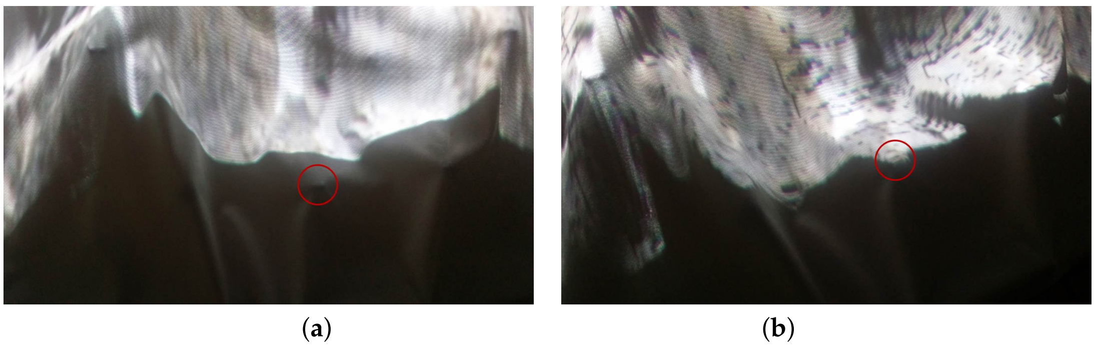

(**a**) (**b**)

Figure 8. Projected texture: (**a**) raw texture image; (**b**) deformed texture image using extrinsic parameters.

The 2D projection coordinates are obtained by multiplying the extrinsic parameter matrix by the coordinates $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ of the curved surface display, measured in real time by the Kinect; these coordinates are then used to deform the raw texture image. However, the Kinect infrared camera has a resolution of 514 × 412, and because the resolution of the table area taken by the Kinect is smaller than that of the Kinect infrared camera, it cannot map all depth value information onto the texture pixel coordinates on a one-to-one basis. Figure 9a displays noise as black spots on the texture image when projecting, after deforming the raw texture image shown in Figure 9b.

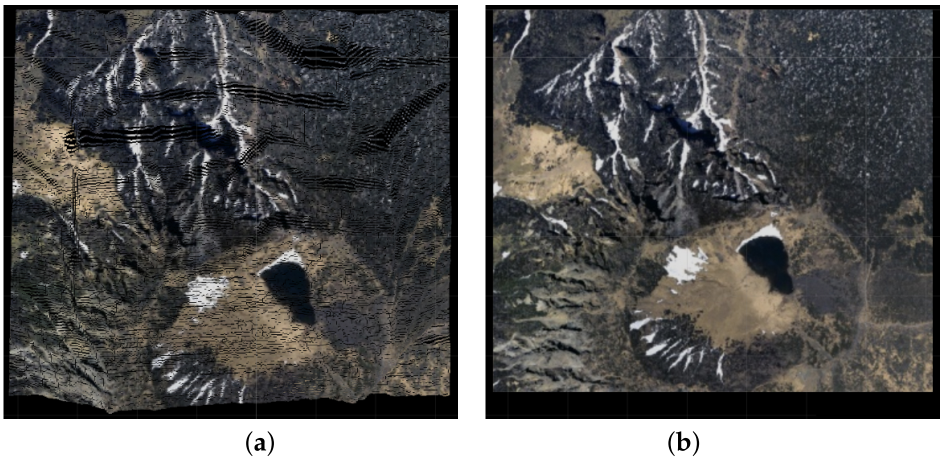


Figure 9. Projection results after texture modification: (**a**) projected texture image; (**b**) raw texture image.

Figure 10a shows an enlarged image of the texture with noise. Textures are divided into pixels with and without color values. A 3 × 3 size box filter is used to remove noise inside this texture [26]. To maintain the shape of the texture outline when noise is removed, color is interpolated only when there are at least four colored areas around the 3 × 3 box filter near the corresponding pixels. Figure 10b shows an interpolated texture image of Figure 9a.

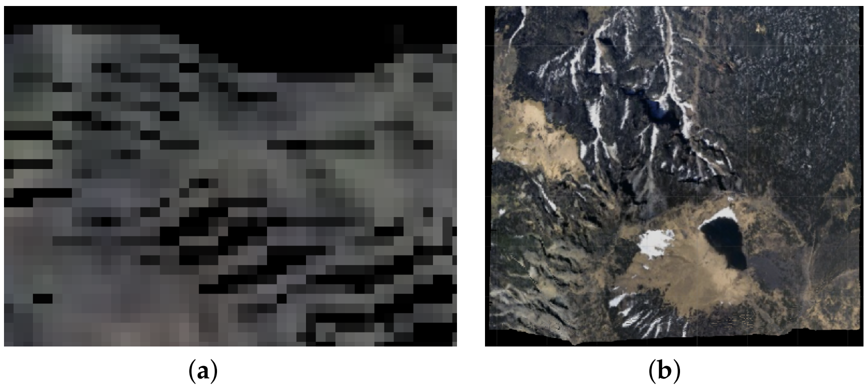


Figure 10. Texture interpolation: (**a**) deformed texture including noise before interpolation; (**b**) interpolated texture using the box filter.

5. User Interfaces/Experiences

The tangible visualization table system supports a gesture-based user interface by using the Kinect installed on the upper part [27]. Figure 11 shows a conceptual diagram of the gesture-based user interface, which divides the space into three separate zones, so giving specific meaning to each space. The actuator space is a space taken up by the curved display, made up of the heights of the linear actuators; this is specified to the maximum height of the actuator (140 mm). The touch interface spaceis a space for selecting the data represented in the Actuator Space as user gestures. The tangible visualization table does not have a touch sensor, but defines a specific distance range from the Kinect to create a touch effect within its space. This space can be used for the selection of important information on the display or parameters to request data from the server. The gesture interface spaceis a space for recognizing general user gestures and is used to control transformation operations such as translating or zooming in/out of data represented in the system; it is also a space to recognize menu activation gestures. The three interaction spaces are not visible to the users, however, the projector displays different icons on the back of the user's hand depending on its height so that the users can interact with the system effectively.

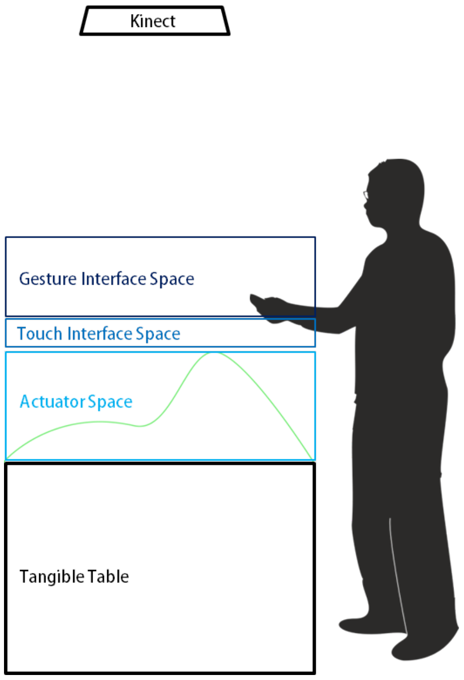


Figure 11. Conceptual gesture interface diagram.

5.1. Gesture Recognition

Gesture recognition is usually designed under the assumption that the user's hand is located within the Touch Interface Space. In order to implement the gesture-based interface with the Kinect, it is necessary to recognize not only the position, but also the shape of the hand [19]. Figure 12 shows a situation in which the hand is recognized by the Kinect RGB, infrared and depth cameras [28]. Figure 12a shows the result of the recognition of the hand by the Kinect RGB camera. When a hand is recognized with an RGB camera, a method of detecting skin color is usually used; however, the hand recognition may not be performed efficiently when the color of the hand is changed by the ambient light. Figure 12b shows the advantage of using an infrared camera to recognize the hand regardless of the light source, but because there is no difference between the surrounding color and the hand color, this is not enough to detect the hand. The stability of gesture recognition is quite crucial for our system. In this paper, we employ a method using the Kinect depth camera. If there are no obstacles between the Kinect depth camera and user's hand, the user's hand shape can effectively be detected by filtering the depth values within a certain distance range. Figure 12c shows the result of hand recognition using the Kinect depth camera. Our method provides high reliability with all users as long as their hands are in the Touch Interface Space. However, our system fails to recognize a ringed hand since a ring causes interference with the Kinect depth camera.

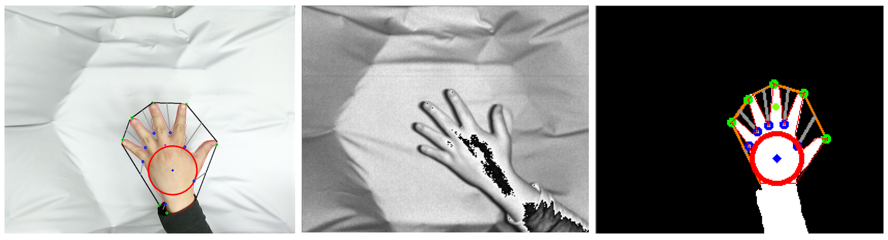

(**a**) (**b**) (**c**)

Figure 12. Comparison of hand recognition: (**a**) RGB camera; (**b**) infrared camera; (**c**) depth camera.

Figure 13a depicts the hand features [29] used for hand recognition, and Figure 13b describes the procedure required to find the hand position. Additional information of each stage is as follows:

- Smooth: Smooth the image for effective processing.
- Find Palm: Find the radius of the largest circle and its center point surrounding the palm in the image (refer to (C_a, r_a) in Figure 13a). Here, the MinMaxLoc function of OpenCV is used.
- Clip Image: Eliminate the part of the image above the wrist in the system gesture recognition process; there is no need for this part, and this reduces the amount of calculation and number of errors at a later stage. The center point C_a and the radius r_a found in the above Find Palm step are used in this calculation.
- Find Contours: Find the outline of the hand using the FindContours function of OpenCV with various options. All of the outline coordinates of the hand for gesture recognition are obtained using the ApproxNone option.
- Find Convex Hull: Find the convex hull including the contour. This is to obtain information to find defects later.
- Find Contour Center Position and Radius: This step finds the radius r_b of the entire hand and its center point C_b. The information found here is used to specify gestures.
- Find Convexity Defects: Find the depth point sitting between the fingertips and the fingers using the contour and convex hull (refer to (p_e, p_d, p_s) in Figure 13a).
- Find Fingers: Find the position of the finger with the information obtained in the above Find Convexity Defects step. Convexity defects contain the coordinates of the depth point between the fingertips and the fingers and the endpoint of the fingertip. The fingertip position can be found using biological features; i.e., that fingers other than the thumb cannot be spread over 90 degrees and the length of fingers is greater than the radius of the palm.
- Find Finger Direction: Compare the position of the end point of each finger found in the Find Fingers step with the coordinates found in the contour. Once the corresponding coordinates are found, the coordinates of the contour are found in the index before and after the appropriate interval around the corresponding index, in the array storing the contour. If the center point of the two identified points and the fingertip point are subtracted, a vector representing the direction of the finger is generated.
- Find Thumb: Even though it is not used in this system, the thumb is searched for to distinguish between the right and left hands. The main features of the thumb are that it is significantly shorter than the other fingers (refer to (l_a) in Figure 13a), the space between the thumb and the index finger is the largest (refer to (c) in Figure 13a) and there is a considerable difference between the length from the tip of the thumb to the depth point and that from the tip of the index finger to the depth point (refer to (l_a, l_b) in Figure 13a).
- Find Gesture: Specify the gesture types based on the data found so far. The gestures used in this paper are shown in Table 3.

Table 3. Gesture shapes.

Gesture Shapes	# of Fingers	Feature	Meaning
Open Palm	4–5	The distance between C_a and C_b is long	A gesture to cancel opening of the menu
Close Palm	0–1	The distance between C_a and C_b is close	A gesture to open the menu
Ok Sign	3	The number of contours is 2	A gesture to transform data displayed on the system
Pointing	1	No thumb	A gesture to select the menu or request data from the server

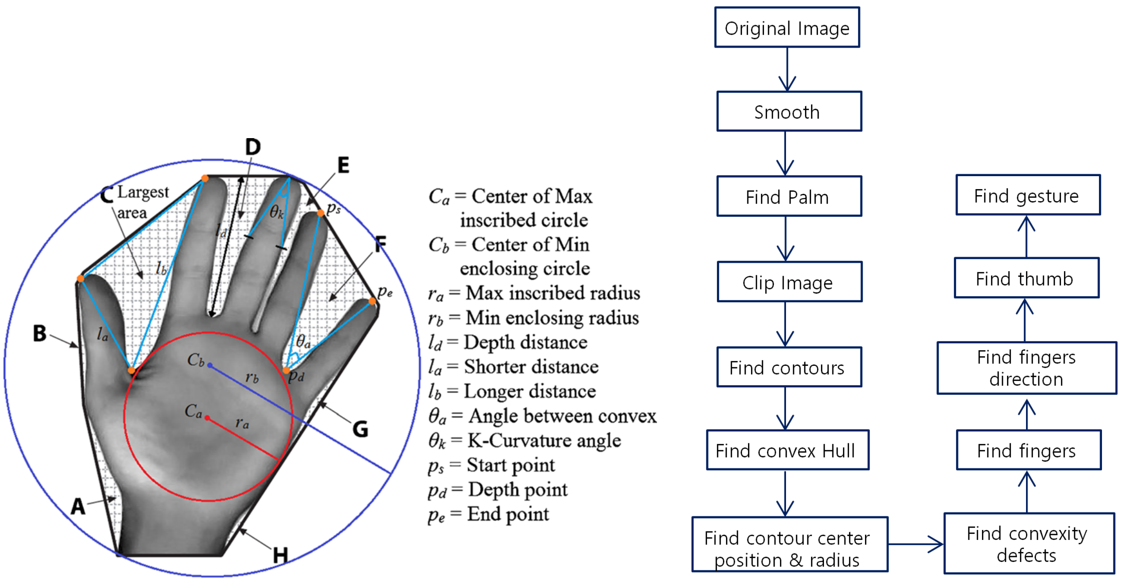


Figure 13. Gesture recognition: (**a**) shape definition for palm recognition; (**b**) gesture interface flowchart.

The system proposed in this paper finds the shapes of the hand and the gesture for each frame according to the above procedure. Figure 14 shows the results for the recognition of input gestures using this process.

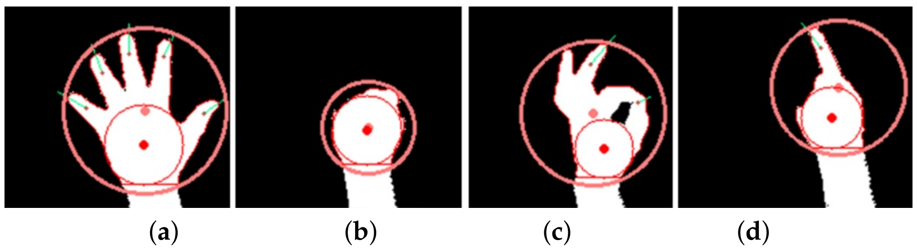


Figure 14. Recognition results of input gestures: (**a**) Open Palm; (**b**) Close Palm; (**c**) Ok Sign; (**d**) Pointing.

5.2. UI/UX Design

Since the interface of this system is based on hand gestures, it should support a UI/UX that takes into account hand shapes, directions and heights. This section explains how each gesture provides the user with an intuitive UI/UX. Figure 15 shows a flowchart for manipulating the gesture-based user interface in this system. The user opens the menu by holding the fist and then selects a particular menu by pointing with a finger. There are three menus: a menu for requesting data from the server, a menu for switching to data transformation and a menu for terminating the program.

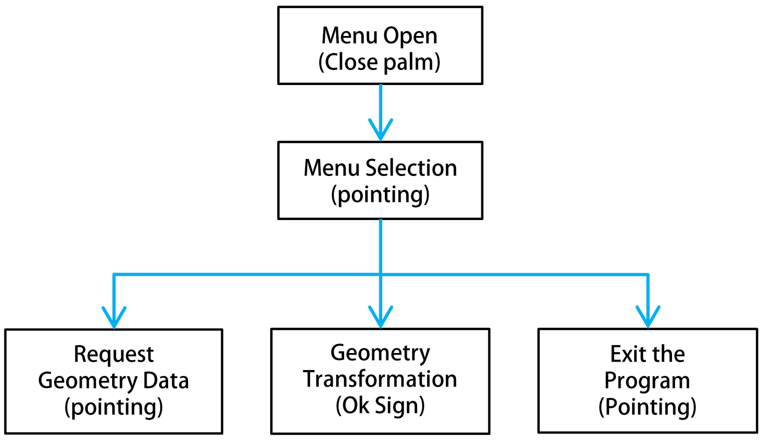


Figure 15. User interface selection flowchart.

5.2.1. Menu Open Interface

An intuitive menu selection is essential for a smooth interaction between the system and the user. There should be no interference with other gestures, and the interface should not run immediately even if the user's hand is recognized. Thus, it is designed to inform the user that the menu has started, and to include a time interval before running the interface in order to confirm the user's willingness to run it.

Figure 16 shows the gesture for menu activation. When the user creates a fist, the projector projects an empty circle on the back of the hand as shown in Figure 16a, and the circle is filled from the inside as shown in Figure 16b. When the circle has been filled, the menu opens, and it takes about two seconds from the moment the circle is initially projected on the back of the hand to the menu opening. During this time, the user can see that the menu is in the middle of opening and will be able to intuitively interact with the system. If the current gesture is not intentional, the user can cancel by creating a different gesture or by retracting the hand from the recognition range of the Kinect. If the fist moves out of position at some speed, the system also tracks the movement to interpret an intention to cancel the menu. Therefore, the user must place his/her hand in a fixed position to open the menu, so that the system recognizes the user's current gesture as an intended action. Once the menu opens, to improve the ease of use, the actuators are then raised to their maximum height; this creates a flat screen and displays the menu and clearly separates the gesture interface space by using all the space taken up by the actuators (see Figure 11 and the video clip at 01:50 s (https://www.youtube.com/watch?v=S2SDb2n4o4M)).

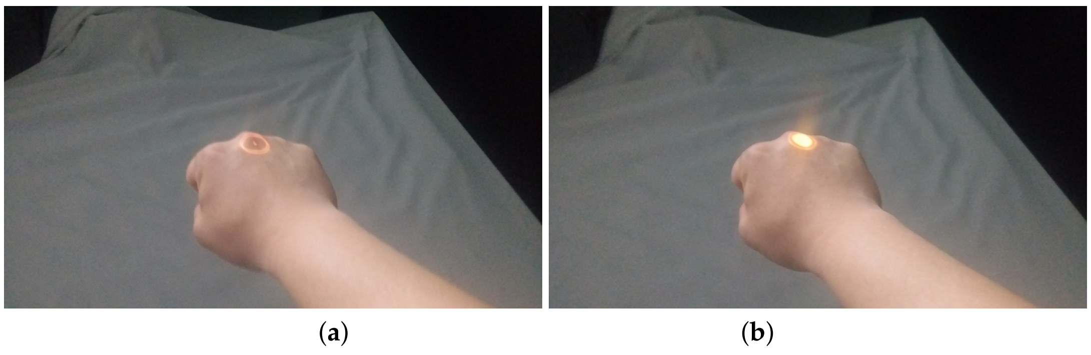


Figure 16. Gesture to open the menu: (**a**) start of menu opening; (**b**) menu opening in progress.

5.2.2. Menu Selection Interface

Figure 17 shows the UI for selecting the next sub-menu when the main menu is open. Each menu icon is placed on a circle around the fist. The user can point his/her finger at what he/she wants to select from the menu. The direction and position information of the finger is calculated using the gesture recognition described in the previous section; i.e., the menu is selected when the angle of the finger direction and the menu direction are calculated and are included in the predetermined angle range (Figure 18). The selected menu icon will change from a muted color to the original bright color and will increase in size to confirm that the menu item has been selected. To see the menu selection interface, readers can refer to the video clip at 02:00 s (https://www.youtube.com/watch?v=S2SDb2n4o4M).

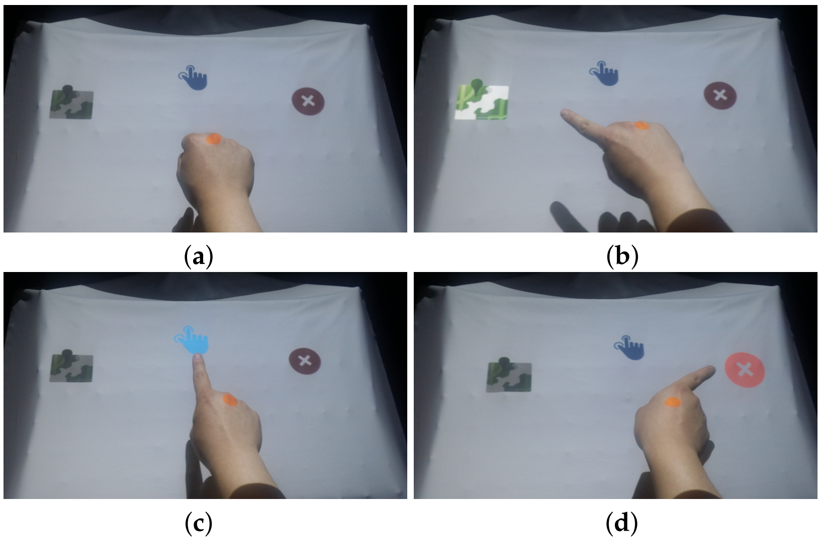

(**a**) (**b**) (**c**) (**d**)

Figure 17. Menu selection: (**a**) menu selection on standby; (**b**) data request menu selection; (**c**) data transformation menu selection; (**d**) program termination menu selection.

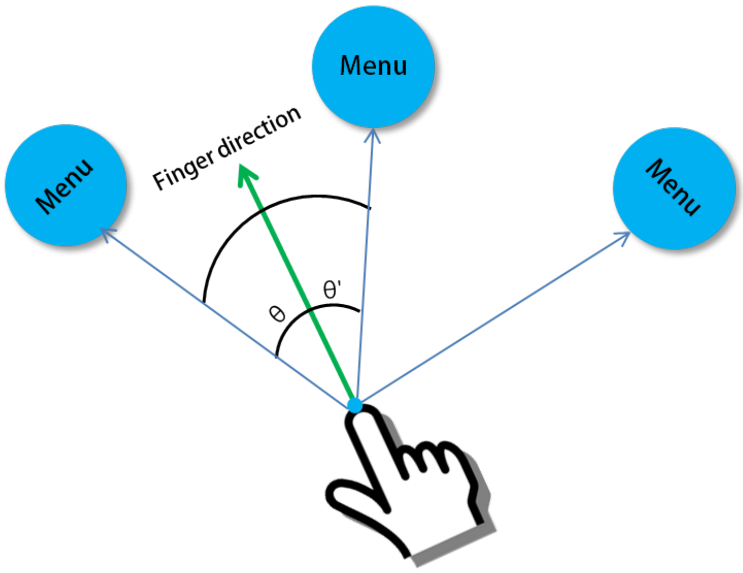
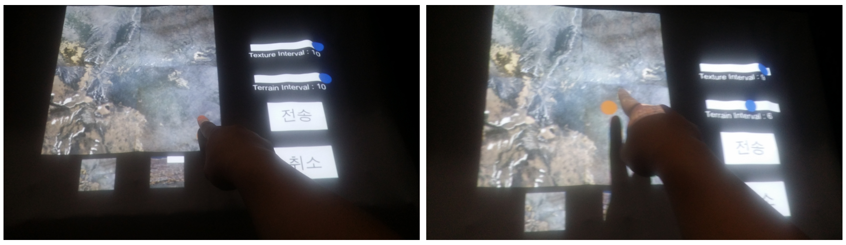


Figure 18. Graphical illustration of menu selection.

5.2.3. Data Request

A data request is through a UI that appears when the menu is selected, as shown in Figure 17b. This UI is for the user to directly request data from the system server (Figure 19). The UI uses the Touch Interface Space shown in Figure 11. The fingertip point of a pointing gesture will be found in this space area, and if the point remains in the corresponding area for a short while, then it will be actioned. As shown in Figure 19a,b, an orange dot follows the finger, which acts like a mouse cursor. It becomes smaller when the tip of the finger stays in the corresponding area, and after a short while, the effect of the touch is applied. These interactions are shown in the video clip at 02:10 s (https://www.youtube.com/watch?v=S2SDb2n4o4M).



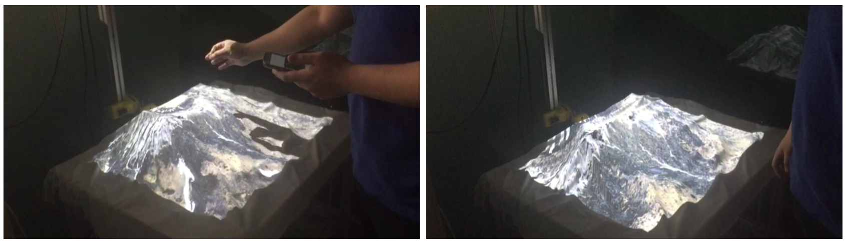
(**a**) (**b**)

Figure 19. Data selection: (**a**) Map Selection 1; (**b**) Map Selection 2.

The largest texture part is used to select a data area, and small panels underneath it are used to select other areas, as shown in Figure 17. When the user selects the area of the texture that he/she wants, a square box appears, and the selected area is marked. On the right are sliders that allow the user to select the LOD (level of detail) of the terrain sampling interval and the texture. This allows the user to select an LOD value ranging from 1–10. When all of the desired data have been selected, data transfer can be requested from the server by pressing the transfer button. The user can also press the cancel button to return to the previous screen.

5.2.4. Data Manipulation

Selecting the Data Transformation menu shown in Figure 17c enables the user to perform transformation operations such as translating or zooming in/out of the terrain data projected on the curved surface display. Figures 20 and 21 show the results of manipulating the visualization part of the terrain data through these operations. The users can make an Ok Sign with their hands and translate the terrain data by moving it in all directions and also zoom in/out of the terrain data by raising and lowering their hand. The transformed terrain is reflected 2 s after the gesture stops. To see the manipulations of terrain data, readers can refer to the video clip at 01:06 s (https://www.youtube.com/watch?v=S2SDb2n4o4M).



(**a**) (**b**)

Figure 20. Terrain data manipulation process (1): (**a**) image of translating terrain data; (**b**) applied image after translating data.

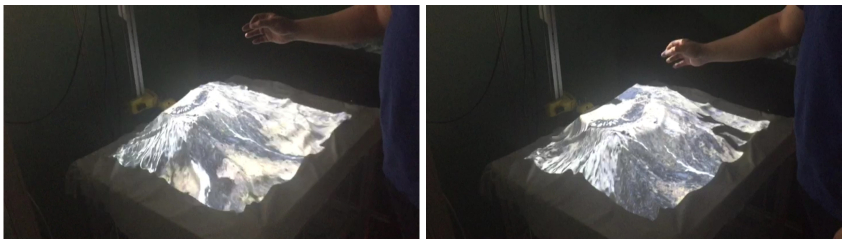

(**a**) (**b**)

Figure 21. Terrain data manipulation process (2): (**a**) before zooming in; (**b**) after zooming in.

6. Experimental Results

In this section, we measure the time taken to generate the curved display in the tangible visualization table and compare the results of other visualization client systems with those of the tangible visualization table. The specifications of the PC used in the tangible visualization table are shown in Table 4. The operating software was developed based on C# using the game engine Unity3d (Version 5.3.3f1) [13]. The Arduino program was implemented using the Arduino IDE (Integrated Development Environment) provided on the Arduino website [12]. The IDE provides an easy development environment with an open source library written in C.

Table 4. Experiment environment.

	Specification
CPU	Intel i7-4790 @ 3.6 GHz
GPU	NVIDIA GeForce GTX 770
RAM	16 GB

The data format used in the system is composed of two types: terrain data (DEM: digital elevation map) and texture data. DEM is a data format that contains real-world topographic information and is used to control the height of actuators. Figure 22 shows all of the texture data used in this system; the user can select a specific area and receive the texture and DEM data of the selected region from the server.

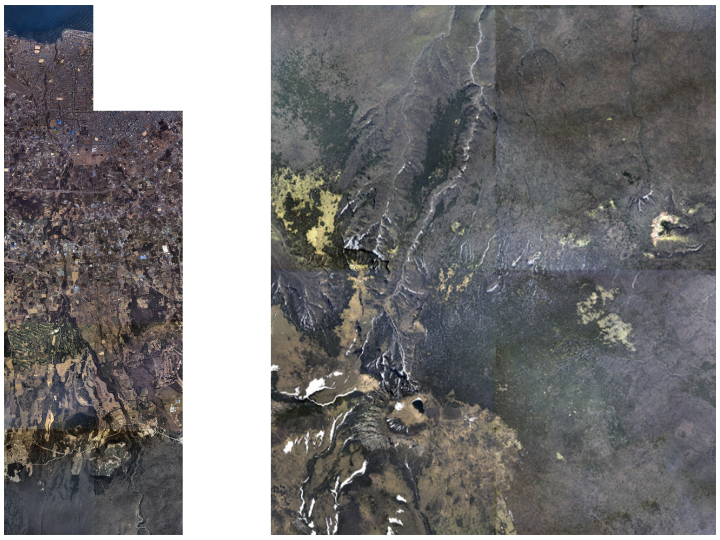

(**a**) (**b**)

Figure 22. Texture data: (**a**) Jeju island; (**b**) Seogwipo.

The system used two types of actuators, one moving at a speed of 8 mm/s and the other at a speed of 32 mm/s. The maximum length of the stroke is 138 mm; the actuator moving at 8 mm/s takes 17.25 s, and the one moving at 32 mm/s takes 4.31 s, when raised from the minimum height to the maximum height. The actuators are raised to the middle height to reduce the time to generate a display when the system is run, and so, the display generation time is cut by half. In addition, high speed actuators are placed intensively in the center of the display, thereby enabling the important part of the display to be created precisely.

The tangible visualization table completes three steps before displaying an image. First, the actuators connected to the screen move to control the flat display; second, the actuators move to generate the curved display; finally, projection mapping and texture interpolation takes place to project the correct image onto the curved display. The first and second steps are affected by the speed of the actuators; the first step takes about 2.1 s, and the second step takes up to 8.6 s. The third step, the projection mapping calculation time, takes 1.25 s on average, as shown in the graph in Figure 23. The projection mapping is calculated after actuator movement stops; it therefore takes a maximum of 12 s to create a curved display.

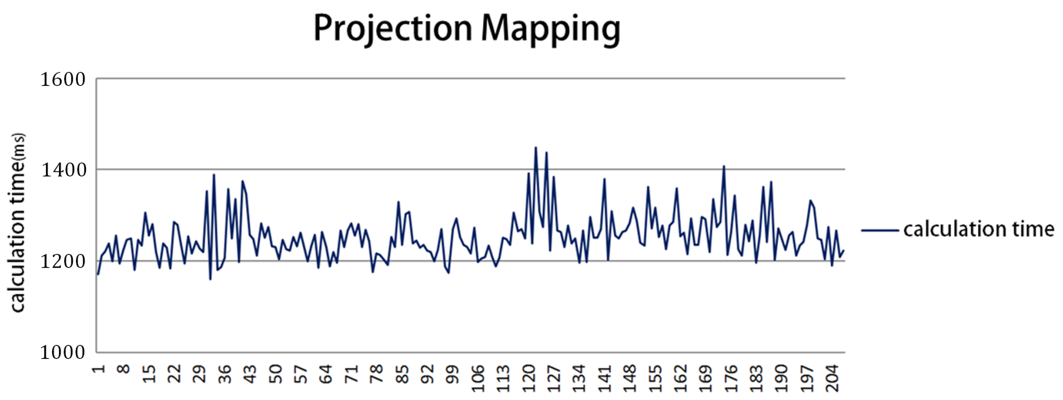
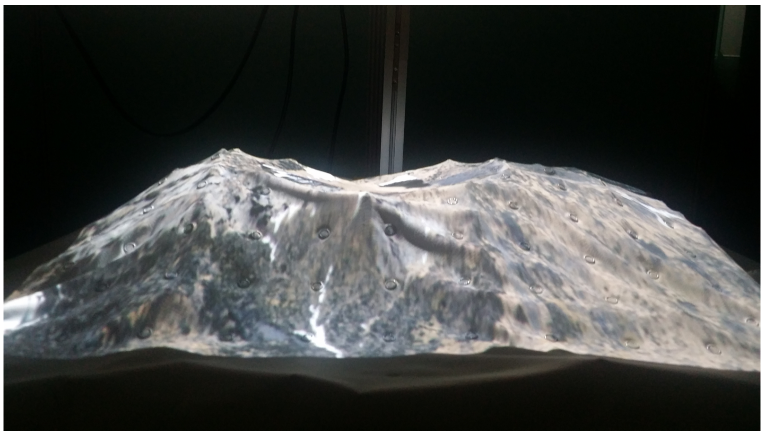


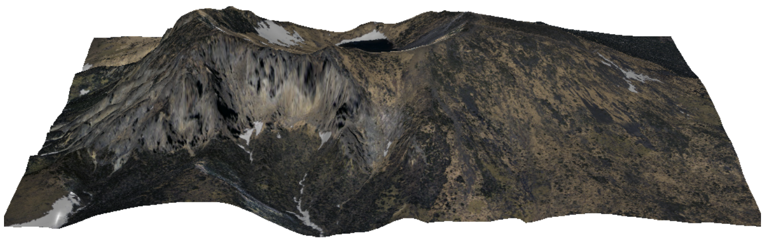
Figure 23. Projection mapping time in milliseconds.

Figure 24a shows the completed display of the tangible visualization table, and Figure 24b shows the visualization results through the monitor display. If the transmission data of the selected area are the same, the visualization image generated in the tangible visualization table is confirmed to be almost identical to that generated in the monitor display.



(**a**)

Figure 24. *Cont.*



(**b**)

Figure 24. Comparison of visualization results between the tangible visualization table and the mobile client: (**a**) tangible visualization table; (**b**) monitor display.

To demonstrate the effectiveness of our system, we conducted a usability test with various participants in Korea Tech Show 2017 (http://www.rndkorea.net/main.asp). All subjects are general users with no experience in this field and aged from their teens to their 40s. We asked the subjects to find a specific area and to understand its topography by using our tangible visualization table. Figure 25 shows the example of target area in the whole region. After finding the target area, the subjects navigated the target area to understand its shape and topography. We gave the subjects 5 min to perform these tasks. After completing the task, we asked them three questions:

- Question 1: Is it intuitive to use the gesture-based interfaces?
- Question 2: Is the visualization table more intuitive and effective than a 2D screen for understanding terrain data?
- Question 3: Do you see any distortions of the projection mapping?

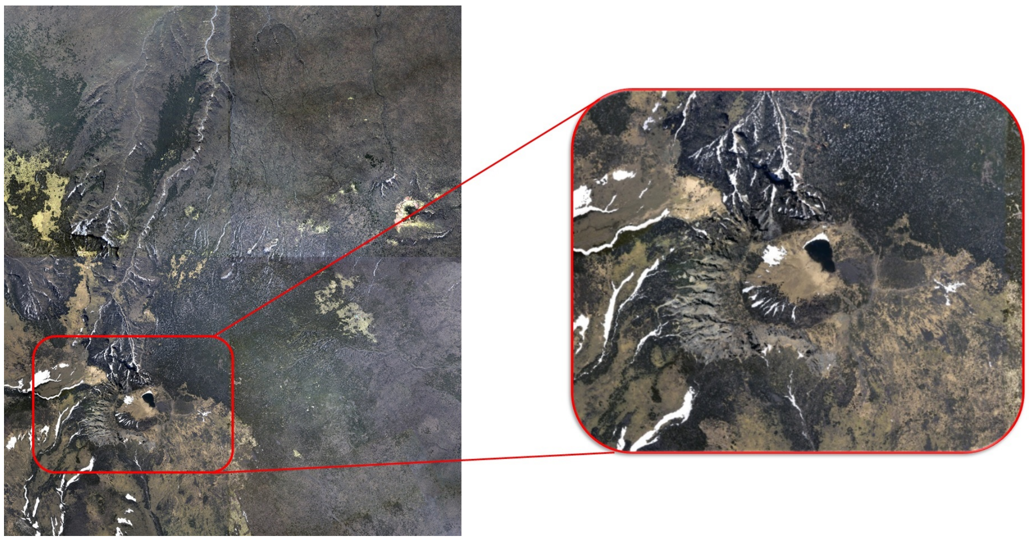


Figure 25. Example of target area in the given task.

They answered with a score between 1 and 5, where 1 is definitely no, 2 is no, 3 is uncertain, 4 is yes and 5 is definitely yes. The mean score of the qualitative evaluation by general users was around 4.3 (yes) for all questions (see Figure 26). In addition to the answers to the questionnaire, some users gave the following feedback:

- All interfaces are easy to learn and use even for non-experts.
- There exists a small delay between user interaction and the reaction of the display surface.
- It would be better to use more actuators for enhancing expressive power.
- It would be better to provide a larger work envelope.

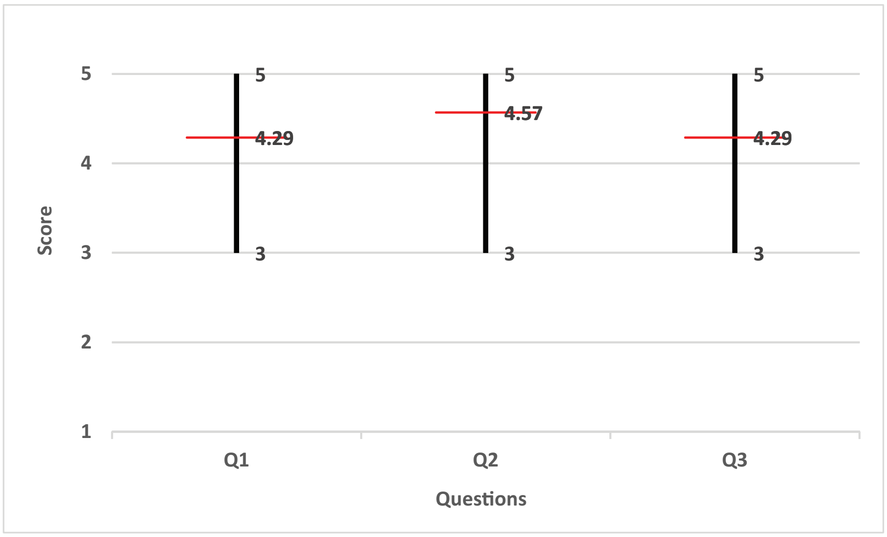


Figure 26. Feedback from users.

7. Conclusions

In this paper, we proposed a tangible visualization table system that can display the terrain data received from a remote server effectively; it does this by controlling multiple actuators and generating the curved display in real time, which can be intuitively touched by the user. A Kinect and projector are calibrated so that textures are projected onto the correct position in the generated display, and LEDs are also employed for realistic visualization. The Kinect-based gesture interface technology enables intuitive control of visualization results. The user can directly request data from a remote server using the gesture interface to enable precise visualization through operations such as translating or zooming in/out. In future research, we would also like to display simulation results of disasters such as typhoons, floods and earthquakes onto the display generated by the tangible visualization table system. In addition, we plan to display more realistic and intuitive data visualization results by using the latest AR equipment, such as Hololens, and developing a more intuitive and easier UI/UX to match the system function. Currently, we are sampling LED textures to represent color values, and in the future, we plan to expand and improve performance to allow visualization of more varied effects, by increasing the number of LEDs and enhancing their functionality.

Acknowledgments: This research was supported by the Korea Institute of Science and Technology Information (KISTI), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03931641).

Author Contributions: Jongyong Kim and Sanghun Park conceived of and designed the experiments. Jongyong Kim and Cheongun Lee performed the experiments. Sanghun Park and Seung-Hyun Yoon wrote the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Boring, E.; Pang, A. Directional flow visualization of vector fields. In Proceedings of the 7th Conference on Visualization 96, San Francisco, CA, USA, 28–29 October 1996; pp. 389–392.
2. Cabral, B.; Leedom, L.C. Imaging vector fields using line integral convolution. In Proceedings of the 20th Annual Conference (SIGGRAPH '93), Anaheim, CA, USA, 2–6 August 1993; pp. 263–270.

3. Lee, D.H.; Kang, M.K.; Yun, T.S. Development of a tangible interface using multi-touch display on an irregular surface. *J. Korea Ind. Inf. Syst. Soc.* **2011**, *16*, 65–72.
4. Leithinger, D.; Follmer, S.; Olwal, A.; Ishii, H. Shape displays: spatial interaction with dynamic physical form. *IEEE Comput. Graph. Appl.* **2015**, *35*, 5–11.
5. Leithinger, D.; Lakatos, D.; De Vincenzi, A.; Blackshaw, M.; Ishii, H. Direct and gestural interaction with Relief: A 2.5D shape display. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, 16–19 October 2011; pp. 541–548.
6. Levoy, M. Efficient ray tracing of volume data. *ACM Trans. Graph. (TOG)* **1990**, *9*, 245–261.
7. Poupyrev, I.; Nashida, T.; Okabe, M. Actuation and tangible user interfaces: The vaucanson duck, robots, and shape displays. In Proceedings of the 1st International Conference on Tangible and Embedded Interaction, Baton Rouge, LA, USA, 15–17 February 2007; pp. 205–212.
8. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87), Anaheim, CA, USA, 27–31 July 1987; pp. 163–169.
9. Reed, S.; Kreylos, O.; Hsi, S.; Kellogg, L.; Schladow, G.; Yikilmaz, M.; Segale, H.; Silverman, J.; Yalowitz, S.; Sato, E. Shaping watersheds exhibit: An interactive, augmented reality sandbox for advancing earth science education. In Proceedings of the AGU Fall Meeting Abstracts, San Francisco, CA, USA, 15–19 December 2014.
10. Wang, C.; Yu, H.; Ma, K.L. Importance-driven time-varying data visualization. *IEEE Comput. Graph. Appl.* **2008**, *14*, 1547–1554.
11. Woodring, J.; Wang, C.; Shen, H.W. High dimensional direct rendering of time-varying volumetric data. In Proceedings of the 14th IEEE Visualization 2003 (VIS'03), Seattle, WA, USA, 19–24 October 2003; pp. 417–424.
12. Arduino. Arduino IDE. Available online: https://www.arduino.cc/ (accessed on 10 December 2017).
13. Unity3d. Unity3d Game Engine. Available online: https://unity3d.com/ (accessed on 10 December 2017).
14. OpenCV. OpenCV Library. Available online: http://opencv.org/ (accessed on 10 December 2017).
15. Leithinger, D.; Ishii, H. Relief: A scalable actuated shape display. In Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction, Cambridge, MA, USA, 25–27 January 2010; pp. 221–222.
16. Piper, B.; Ratti, C.; Ishii, H. Illuminating clay: A 3D tangible interface for landscape analysis. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Minneapolis, MN, USA, 20–25 April 2002; pp. 355–362.
17. Hilliges, O.; Izadi, S.; Wilson, A.D.; Hodges, S.; Garcia-Mendoza, A.; Butz, A. Interactions in the air: Adding further depth to interactive tabletops. In Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, Victoria, BC, Canada, 4–7 October 2009; pp. 139–148.
18. Follmer, S.; Leithinger, D.; Olwal, A.; Hogge, A.; Ishii, H. inFORM: Dynamic physical affordances and constraints through shape and object actuation. In Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, St. Andrews, UK, 8–11 October 2013; pp. 417–426.
19. Yeo, H.S.; Lee, B.G.; Lim, H. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimed. Tools Appl.* **2015**, *74*, 2687–2715.
20. Mine, M.R.; van Baar, J.; Grundhofer, A.; Rose, D.; Yang, B. Projection-based augmented reality in Disney Theme Parks. *Computer* **2012**, *45*, 32–40.
21. Herrera, D.; Kannala, J.; Heikkilä, J. Accurate and practical calibration of a depth and color camera pair. In Proceedings of the International Conference on Computer Analysis of Images and Patterns, Seville, Spain, 29–31 August 2011; pp. 437–445.
22. Weng, J.; Cohen, P.; Herniou, M. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 965–980.
23. Velizhev, A. GML C++ Camera Calibration Toolbox. Available online: http://graphics.cs.msu.ru/en/node/909 (accessed on 10 December 2017).
24. Shah, S.; Aggarwal, J. Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation. *Pattern Recognit.* **1996**, *29*, 1775–1788.
25. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes in C*; Cambridge University Press: Cambridge, UK, 1992.

26. Malvar, S.; He, L.W.; Cutler, R. High-quality linear interpolation for demosaicing of bayer-patterned color images. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, QB, Canada, 17–21 May 2004; pp. 485–488.
27. Pavlovic, V.I.; Sharma, R.; Huang, T.S. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 677–695.
28. Bretzner, L.; Laptev, I.; Lindeberg, T. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition, Washington, DC, USA, 21 May 2002; pp. 423–428.
29. Bradski, G.; Kaehler, A. *Computer Vision with the OpenCV Library*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2008.