V. Kůrková et al. (Eds.): ITAT 2014 with selected papers from Znalosti 2014, CEUR Workshop Proceedings Vol. 1214, pp. 34–39
http://ceur-ws.org/Vol-1214, Series ISSN 1613-0073, © 2014 R. Gargalík, Z. Tomori

Control of Depth-Sensing Camera via Plane of Interaction

Radoslav Gargalík[1,2] and Zoltán Tomori[2]

[1] Inst. of Computer Science, Faculty of Science
P. J. Šafárik University in Košice, Slovakia
radoslav.gargalik@gmail.com
[2] Inst. of Experimental Physics
Slovak Academy of Sciences, Košice, Slovakia
tomori@saske.sk

Abstract: Depth-sensing cameras (e.g. Kinect or Creative Gesture Camera) are exploited in many computer vision and augmented reality applications. They can also serve as a key component of natural user interaction via virtual keyboard, body pose or hand gestures. We integrated both these functions proposing the "Plane of Interaction" (POI) which is a solid flat surface placed on the reference plane (table top, floor). Calibrated camera/projector system automatically identifies the position of POI surface, projects the virtual menu buttons onto it and recognizes which button was "clicked" by hand (fingertip).
The proposed POI was tested with the camera/projector prototyping setup. POI allows natural and quite robust interaction in this specific environment.

1 Introduction

Human-computer interaction is one of the most progressive research areas of computer science because it simplifies the control of electronic devices and opens the door for new potential users. Natural User Interface (NUI) represents the latest stage trying to exploit gestures, voice commands, gaze tracking, brain computer interface based on the analysis of EEG signals etc. Probably the most popular NUI outcome are touch gestures applied within last few years into the touch tablets and smart phones.

Low-cost depth-sensing camera Microsoft Kinect [11, 12] launched in Nov. 2010 opened a new era of 3D computer vision applications. Color camera combined with the depth sensor dramatically simplifies some computer vision algorithms like e.g. segmentation of 3D scene. Easy segmentation of the human body or hand allows calculation of the 3D coordinates of corresponding skeletons. This representation simplified the recognition of hand gestures like "wave", "circle", "swipe", "pinch" etc. Some new 3D cameras with built-in gestures recognition capabilities like "Creative Senz3D Camera" [14] or "Leap Motion" [13] appeared recently. Open source libraries e.g. OpenNI and OpenCV [15] support the broader range of such cameras so applications like "virtual keyboard", "air harp" and many others are available via internet. On the other hand, very quick progress resulted in missing official standards in this area. Interpretation of some gestures or the other forms of interaction can be natural in some application but they are quite confusing in some others.

Projection-based augmented reality [3, 4] projects images onto the real surface using one or several projectors. Depth-sensing camera can make such applications interactive – user can change the shape or the position of the real objects which is followed by the change of projected color, image, animation etc.

In addition to the interaction with an object, sometimes is required also the interaction controlling the program itself (e.g. change the mode of operation). The use of a mouse or a keyboard would be complicated and unnatural in this situation. Therefore we created the specific object (plane of interaction) which is a natural part of augmented reality environment but its function is to control the program via the projected virtual buttons.

2 Related Work

There are many application areas, in which a pair Kinect/projector can be used for augmented reality. Many researchers developed different augmented/virtual reality graphical user interfaces over the time. They differ in different hardware requirements (such as haptic pens, special virtual glasses, etc.) and features.

Szalavari in his dissertation [6] proposed the augmented reality panel called *Personal Interaction Panel* (PIP). PIP consisted of a black board (as a panel), a haptic pen and a head mounted display. The panel and the pen were tracked with *Polhemus Fastrak* (six degree-of-freedom) tracker, where the receiver was mounted to the head mounted display. As the head mounted display the *Virtual I/O i-glasses!* was used. The base principle of this approach is, that electomagnetic tracker tracks the panel and haptic pen. The head mounted display is then used to overlay graphics onto the real environment (mainly the panel). Electromagnetic emitter and receiver worked at 30 Hz. The disadvantage of this approach is additional hardware requirements (haptic pen, head mounted display and electromagnetic emitter/receiver).

Poupyrev et al. in [7] proposed the concept of a *Generic Augmented-Reality Interface*. They used *tiles*, which are printed paper cards (15×15 cm each) with simple square patterns consisting of a thick black border and unique symbols in the middle. According to authors, any symbol can be used for identification. There are two type of *tiles*: physical icons and *phicons*. *Phicons* propose a close

coupling between physical and virtual properties so that their shape and appearance mirror their corresponding virtual object or functionality. The user can freely manipulate with tiles, which is also a default way of interaction with real objects represented by tiles. User must wear a lightweight Sony Glasstron PLMS700 head-set. Main steps of this approach include tracking rectangular markers of known size, calculating the relative camera position and orientation in real-time, and finally rendering virtual objects on the physical paper cards. The system runs at 30 FPS and was implemented with the open-source ARToolKit software library. Although this concept is interesting and the developed generic augmented-reality interface can be used in many practical ways, there is also the same disadvantage as in the previous approach – need of special head mounted display.

The similar approach was used by Geiger et al. in [8] to construct the *ARGUI* augmented-reality system. The system was constructed with utilization of ARToolKit, OpenGL and GLUT libraries. Depending on the way the 2D cursor is positioned on the augmented reality pattern, two modes are available in the system: cursor based and marker based interaction. Cursor based movement means that the 2D mouse cursor is moved using a suitable input device (such as mouse or tablet). Marker based movement means that the video camera is moved to position the augmented reality pattern-object under the "static" mouse cursor. A mixture of both modes is also possible. In real application (augmenting real paintings with additional information about artist, painting techniques, historical information and other important information), a head mounted display (Eyetrek) was used with a mounted USB camera (Phillips ToUcam). To control the cursor, a remote control with gyrotechnology was used. Again, as in all previous approaches, additional hardware (head mounted display and GyroControl) must be used.

Benko et al. in [9] presents a *Projected Augmented Reality Tabletop*. One of the system features is a freehand interaction. The system consists of the Kinect depth-sensing camera, stereo projector (Accer H5360), shutter glasses (Nvidia 3D vision), stereo sync emitter (Nvidia 3D vision) and a table. The Kinect scans objects before the table, so they can be projected as a mirror view. To provide correct 3D perspective view of the virtual scene, the user's head location and gaze must be tracked. To track the user's head, the disturbing reflectivity of the shutter glasses is used. The reflectivity creates "holes" in the acquired depth map, so the aggregate location of those holes can be tracked. Freehand physically realistic interactions are simulated using a commercial Nvidia PhysX game engine.

3 Plane of Interaction

Our main motivation was to find a simple and easy way how to interact with the program during experiments with augmented reality environment using the depth sensor instead of the mouse or keyboard. Proposed "Plane of Interaction" (POI) is a part of scene and exploits the same depth-sensing camera as the basic program. From the technical point of view, POI is a planar surface which can be placed anywhere inside the field of view of the camera, it should be automatically identified by the camera and exploited as a virtual touch sensor.

In this section we will describe how to acquire, calibrate and extract the important information using the POI. In this chapter we will describe how to acquire, calibrate and extract the important information using the POI.

3.1 Mechanical Parts

For prototyping purposes and for experiments with projector-based augmented reality we constructed the setup shown on Fig. 1. Projector *P* and 3D camera *C* share the same mount attached to the massive stand. The translation and possible rotation between them compensates calibration software (see subsection 3.4 – *Calibration*). Plane of Interaction is a solid planar plate, which can placed anywhere inside the field of view of the camera.

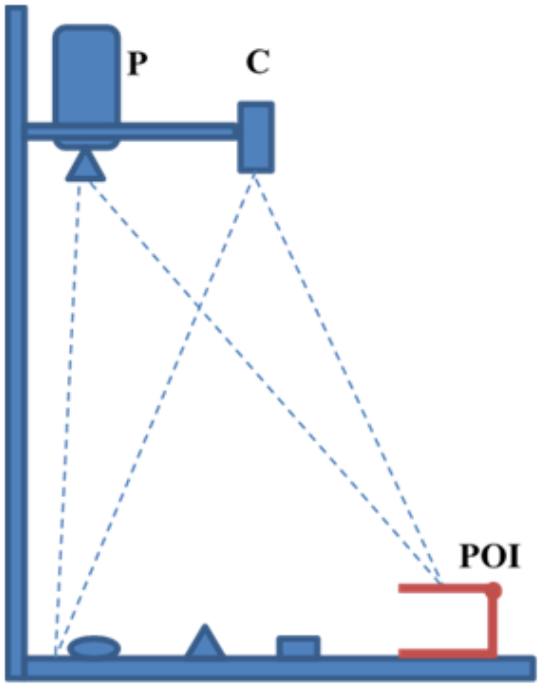

Figure 1: Setup for Projection-Based Augmented Reality Experiments. *P* – Projector, *C* – 3D Camera, *POI* – Plane of Interaction.

3.2 Hardware and Software

We exploited projector BENQ MX613ST (aspect ratio 4:3, through ratio cca. 1.0) and Microsoft Kinect [11] depth-sensing camera. We used OpenNI library to control the camera acquisition process. It offers a set of functions to

acquire RGB and depth images and basic functions to process them.

3.3 Reference Plane Detection

We can imagine the reference plane as a sea level where the height of all objects is measured as a distance from it. POI is oriented parallel to the reference plane (floor, table top). Despite the precise adjustable mounting we cannot guarantee that both Kinect and the projector are perpendicular to the floor. Therefore we acquire the background image which is the image of flat surface (floor, desktop) without any objects placed on it. Then we fit the background image by the analytical equation of the plane using the RANSAC algorithm [1]. From the resulting analytical solution we generated again the background image which will be subtracted from every captured depth image.

3.4 Calibration

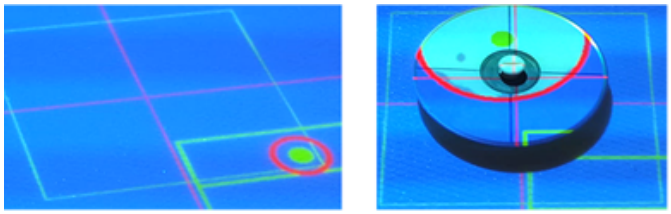

Figure 2: Part of the Calibration Process. Projected Grid to the Surface (Left) and a Small Disc Placed Into the Grid Intersection (Right).

The camera and the projector have different poses (translated and rotated relative to each-other). They have the different resolution and we have to take into account also the recalculation of pixels to length units (millimeters). All of these problems should be solved by the following geometrical transformation

$$x_P = T_x(x_C, y_C, z_C) \qquad y_P = T_y(x_C, y_C, z_C) \tag{1}$$

where T_x and T_y are linear transformations. It means that each 3D point (x_C, y_C, z_C) measured by the 3D camera is transformed into the 2D point (x_P, y_P) displayed by the projector. This problem is similar to "Bundle adjustment" method in [1]. The way how to determine transformation functions T_x, T_y is to find corresponding pairs of camera points V_C and projector points V_P and minimize the reprojection error (solve the least square problem). Let us denote $V_C = (x_C, y_C, z_C, 1)^{\mathrm{T}}$ as the point measured by the 3D camera and $V_P = (x_P, y_P)^{\mathrm{T}}$ as the same point, which is projected by the projector. Assuming that the difference between position and orientation of the 3D camera and the projector can be expressed by rotation and translation, we can first translate the point V_C from the 3D camera coordinate system to the 3D orthonormal coordinate system of the projector, where the point $[0,0,0]$ is inside the projector and the z-axis is in the direction of the projection.

$$V_{P0} = AV_C \tag{2}$$

$$\begin{pmatrix} x_{P0} \\ y_{P0} \\ z_{P0} \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} \tag{3}$$

To transform the point from the 3D coordinate system of the projector ($V_{P0} = (x_{P0}, y_{P0}, z_{p0})^{\mathrm{T}}$) to the 2D coordinate system of the projector ($V_P = (x_P, y_P)^{\mathrm{T}}$), we can use the following equations

$$x_P = \frac{c_1 x_{P0}}{z_{P0}} \qquad y_P = \frac{c_2 y_{P0}}{z_{P0}} \tag{4}$$

where c_1 and c_2 are unknown coefficients related to the ratio of the projector and scaling from millimeters to pixels. Further expressing of equations 4 and modification of matrix A leads to a system of 11 linear algebraic equations (more details can be found in[2]). If we enter more pairs of corresponding points (V_P, V_C) than the number of equations is, then we obtain an overdetermined system which can be solved by the QR matrix decomposition. The resulting matrix contains the coefficients describing the transformation between the Kinect 3D camera and the projector coordinates.

In practice, we need cca. 30 pairs of corresponding points to achieve reasonable precision. We have developed a special program which simplifies their acquisition. We project a grid to the surface and place a small disk into all grid intersections (see Fig. 2). The program on each intersection automatically acquires the depth image of the disk, finds its contour and fits the contour by the model of an ellipse. The center of such a disk determines coordinates (x_C, y_C) and the height of the disc above the reference plane is z_C. Coordinates of the projector points (x_P, y_P) are the known grid intersections.

As the positions of the camera and the projector are stable, the calibration should be performed only once. Then the calibration data are saved as a part of the configuration file. The calibration coefficients are the same for the POI rectangle and for the rest of the depth image.

It should be noted, that to obtain optimal precision of the transformation, not all pairs of corresponding points should lie in one plane. If all pairs of corresponding points lie in one plane, then the error raising from inaccurate transformation between the 3D Kinect camera and the projector raises with the absolute distance of the measured 3D points from the plane in which pairs of corresponding points were acquired. In that case some serious inaccuracy can be seen on the image projected to the POI surface if the POI surface (plane) is not quite near to the plane in which the pair of corresponding points lie.

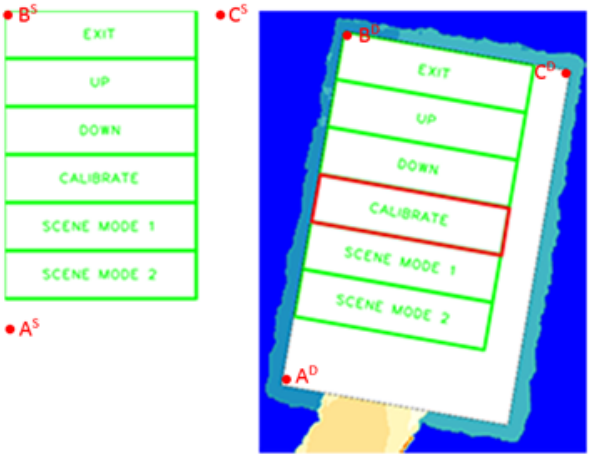

Figure 3: In-memory Image of Menu Buttons (Left) and Projected Menu Buttons Onto the POI Surface (Right).

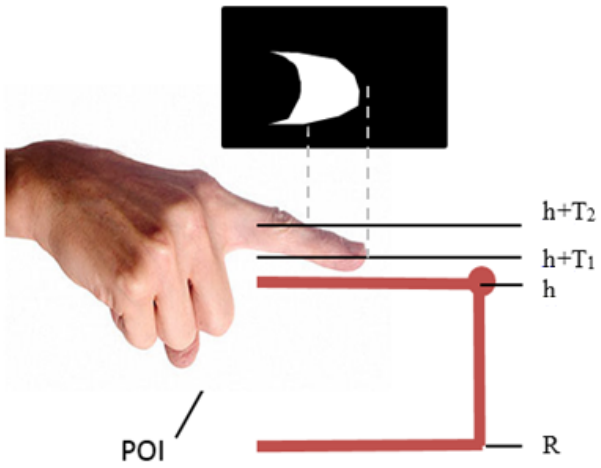

Figure 4: Detection of the Fingertip.

3.5 Detection of Finger Position

The image of menu buttons is projected onto the surface of POI (see Fig. 3 Right). The goal is to identify the button rectangle touched by a fingertip.

It should be noted, that menu buttons should be projected onto the surface regardless of the orientation of the surface (see Fig. 3 Right). So if the surface is rotated, then we must ensure proper orientation of the menu buttons, too.

The first step in our approach is to detect POI surface. After that we try to find the smallest possible rectangle (minimal area), which will contain all points from POI surface. We also calculate its rotation.

Once this is known, we calculate the affine transform between in-memory image, which is not rotated and the projected image, which is rotated according to the calculated angle. Let us denote this affine matrix as M. To calculate M we need three 2D corresponding points between in-memory image and projected image. Those points are depicted on Fig. 3 as points A, B and C, where superscript S denotes source image and superscript D denotes destination image. Clearly, all six points (A^S, B^S, C^S, A^D, B^D, C^D) can be calculated automatically.

Now the depth sensor watches the hand above the ROI surface from the top. The reference plane (see Fig. 4) is labeled R, the POI is h units above the R. Two threshold levels T_1 and T_2 represent the range of sensitive distances from POI. 3D points having z-coordinate (depth) from the interval $<h+T_1, h+T_2>$ create a binary image representing a rough approximation of the fingertip (in our experiments we used $T_1 = 5$ and $T_2 = 20$ millimeters). As stated above, the POI surface can be rotated, so to find out which button is "clicked", we first transform the binary image of the fingertip approximation to the in-memory menu image coordinates using the inverse transform of matrix M. The result of this step is depicted on Fig. 5 (Middle). To eliminate noise (the outline of the hand and other fingers), we apply the OPEN morphology operation to the binary

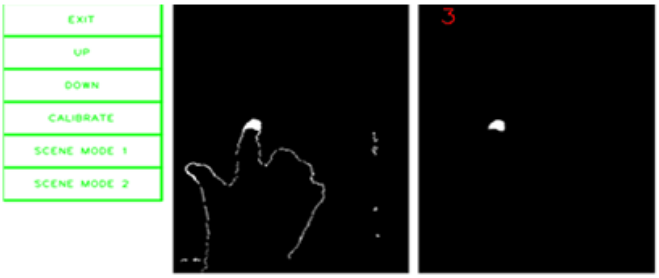

Figure 5: In-memory Image of Menu Buttons (Left), Binary Image of Detected Fingertip with Noise (Middle) and Detected Fingertip After OPEN Morphology Operation Performed and with the Zero-based Index of the "Clicked" Virtual Button (Right).

image. In our experiments we used 3×3 rectangle as a kernel with an anchor point in the center of the rectangle and OPEN morphology operation was applied in two iterations. The difference can be seen on Fig. 5 (Middle and Right). To finally find out which virtual button is "clicked", we count non-zero pixels in each virtual button rectangle from the binary image and we select the one, which has the maximum number of white pixels in its rectangle. To eliminate recognition of some small region as a fingertip, we use another threshold value T_f and we "select" some virtual button only if the counted white pixels exceed the threshold value T_f.

In our experiments we used $T_f = 150$ pixels. In case we expect some smaller fingers (such as fingertips of children), the threshold value T_f should be set to something between 70 and 100 pixels.

It should be noted that the orientation of the hand is not critical to this approach, which is clearly an advantage. It is possible to place the hand from any direction to the POI surface and as we stated previously, the POI surface can be freely rotated on the reference plane.

4 Augmented Reality Sandbox

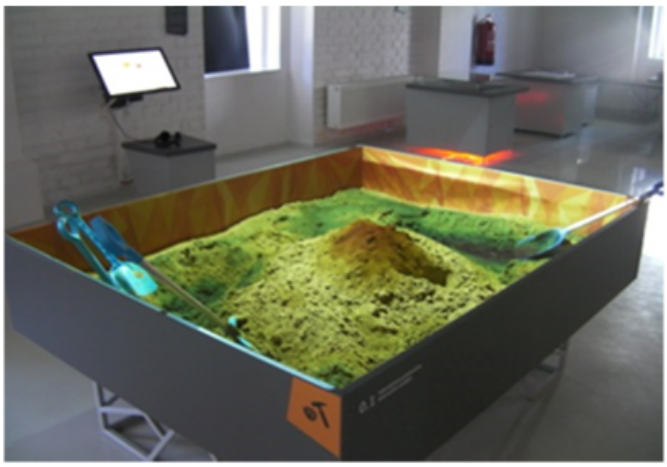

Figure 6: Augmented Reality Sandbox. The Color Projected to Sand Depends on the Height.

Science museum (called also "Theme Park", "Science center", "Discovery Center") is a modern type of museum where most of exhibits are interactive. One such museum was opened last year in our city where our group participated in the construction of several exhibits. Augmented reality sandbox is one of them, which exploits the depth-sensing camera.

The construction of our interactive sandbox was inspired by [4]. Both, Kinect camera and projector are attached to the ceiling above the box with sand (see Fig. 6). Kinect measures the elevation of sand terrain and the projector illuminates the sand by the corresponding colors (hills by brown color, lakes by blue one etc.). The change of terrain is followed by the change of color with minimal delay. The table defining colors for individual heights intervals can be defined by the user. The sandbox was calibrated by the same algorithm as described in the subsection 3.4 – *Calibration*.

4.1 Specifics of User Interaction in the Science Museum

Conditions in the science museum are very specific:

- Most of visitors are groups of children requiring simple, robust and self-explanatory control of the exhibits.
- No extra hardware such as keyboard, mouse, cables etc. is acceptable.
- The virtual menu should exploit the same camera as the exhibit itself.
- The function of menu must be very simple – usually just select the mode of operation.
- Periodical innovation and upgrade of exhibits is the necessary condition to achieve repeating visits of the same people.
- The museum is open daily for more than 100 of visitors per day so the time for the installation and testing of exhibits in real conditions is limited.

From the beginning, our sandbox operated only in a single mode as described above. However, we plan to implement new features and modes in the near future. POI concept described in this paper allows the easy way how to switch between various modes of operation by "clicking" the virtual buttons by hand.

5 Results

We proposed a simple system of interaction with program exploiting depth-sensing camera called "Plane of Interaction". POI is the integral part of virtual reality environment with specific function – recognize the position of finger (fingertip) and return the index of "clicked" rectangle representing the virtual button.

The proposed system works in real-time. In our experiments we achieved more than 30 frames per second, which is enough for real-time augmented reality applications. In fact, because the FPS of the Kinect depth-sensing camera is 30, more than 30 FPS is not needed.

We tested POI on the prototyping system with satisfactory results. Currently, we test POI in real conditions of the science museum with augmented reality sandbox exhibit.

6 Future Work

The augmented reality sandbox installed in the science museum was continuously tested by real visitors. Although the anonymous survey declared mostly high and very high satisfaction of visitors, practical experience revealed some problems and showed new possible improvements. The calibrated Kinect/Projector/Sandbox system can be easily extended to create other augmented reality applications.

Anyway, technological aspects are only one side of the coin because they must be in balance with other exhibits in the science museum. Therefore any significant changes in exhibits must be consulted with other authors and designers.

The functionality of the POI can also be extended, so it can be used in the similar way as default touch screens and displays widely used today. We plan to extend the functionality of the POI concept described in this paper with the following features:

- Multi-select. This feature enables users to use more than one finger to perform a multi-select. It can be

utilized to selected more objects at once or, for example, to "check" multiple virtual check-boxes.

- Select and move. This feature is very similar to the widely used drag & drop feature and enables users to select some object (or with the combination with the previous features to select multiple objects at once) and to move it somewhere else. The "select" part of this feature will be exactly the same concept as the one described in this paper, which is similar to the mouse down event. To simulate mouse up event, we just check for moving the fingertip (or fingertips) away from the POI surface.

- Recognition of basic gestures. If it is possible to simulate "click and move", then we can use the points obtained during the "move" phase as a gesture and try to recognize it. Basic gestures (such as swipe fingertip left/right/up/down) can be recognized quite easily. To recognize more complicated gestures one can use Dynamic Time Warping algorithm [10] or utilize some machine learning approach.

- The recognition of basic drawing shapes. With the aid of the "select and move" feature, we can enable users to draw some basic shapes (such as a line, circle, rectangle, etc.). To recognize such drawn shape, we can use pattern matching or in case of simple shapes (such as triangle, rectangle, circle, etc.) we can find a contour of drawn shape and try to fit the contour with a polygon. After that we can count the number of vertices of the polygon to recognize this basic shape.

It should be noted, that not all improvements listed above are directly connected to the augmented reality sandbox described in section 4. Some of the improvements are planned to be used in the future in our other projects.

7 Conclusion

The plane of Interaction is the alternative to control the exhibits in science centers with the majority of children visitors. It is simple, self-explanatory and robust. For interaction it does not require any extra hardware only this included in the exhibit.

Acknowledgment

This work was supported by Slovak research grant agencies APVV (grant 0526-11).We thank to US Steel Košice as the main contributor of Steel Park science museum and "City of Košice" as co-founder.

References

[1] R. I. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*, 2nd ed. Cambridge University Press, 2004.

[2] J. Hrdlicka, *Kinect-projector calibration, human-mapping*, 3dsense interactive technologies blog, 2013. http://blog.3dsense.org/programming/kinect-projector-calibration-human-mapping-2/

[3] M. Mine, D. Rose, B. Yang, J. Vanbaar and A. Grundhofer, *Projection-Based Augmented Reality in Disney Theme Parks*, Computer 45, 7, 32-40. 2012.

[4] O. Kreylos, *Augmented Reality Sandbox*, UC Davis. 2013. http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/

[5] Z. Tomori, R. Gargalik and I. Hrmo, *Active Segmentation in 3D using Kinect Sensor*, In Proceedings of the 20th International Conference on Computer Graphics, Visualization and Computer Vision '2012 (WSCG 2012), Part 2 (Pilsen, Czech Republic, June 26-29, 2012, 2012). Vaclav Skala Ed., Pilsen.

[6] Z. Szalavari and M. Gervautz, *The Personal Interaction Panel – a Two-Handed Interface for Augmented Reality*, Computer Graphics Forum, pp. 335-346, 1997.

[7] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht and N. Tetsutani, *Developing a GenericAugmented-Reality Interface*, Computer (Volume 35, Issue 3), pp. 44-50, 2002.

[8] Ch. Geiger, L. Oppermann and Ch. Reirnann, *3D-Registered Interaction-Surfaces in Augmented Reality Space*, In Augmented Reality Toolkit Workshop, pp. 5-13, 2003.

[9] H. Benko, R. Jota and A. D. Wilson, *MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop*, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 199-208, 2012.

[10] S. Salvador and P. Chan, *Toward Accurate Dynamic Time Wrapping in Linear Time and Space*, In Intelligent Data Analysis, 11(5):561-580, 2007.

[11] Microsoft, *Kinect for XBox 360*. http://www.xbox.com/en-US/kinect

[12] Microsoft, *Kinect for Windows*. http://www.microsoft.com/en-us/kinectforwindows/

[13] Leap Motion, Inc., *Leap Motion*. https://www.leapmotion.com/

[14] Creative Technology Ltd., *Creative Senz3D*. http://us.creative.com/p/web-cameras/creative-senz3d

[15] Willow Garage and Itseez, *OpenCV library*. http://opencv.org/