Realization of Natural Interaction over the Micro Soot Device Model

Petar Mrazović, Marko Pilipović, Mario Volarević, Željka Mihajlović
Faculty of Electrical Engineering and Computing, Zagreb, Croatia
petar.mrazovic@gmail.com, marko.pilipovich@gmail.com, mario.volarevic@gmail.com, zeljka.mihajlovic@fer.hr

Abstract - Paper explores methods of achieving natural interaction between the virtual and the real world, along with their application in industry. Significant emphasis is given to the Microsoft Kinect device, whose functionality was introduced in the field of powertrain systems industry. Collaboration with Austrian company AVL, which is engaged in development, testing and simulation of powertrain systems, has contributed with specific problem that can be solved in augmented reality domain. In order to facilitate maintenance of special device for continuous measurement of lowest soot concentration in the diluted exhaust from internal combustion engines, an interactive virtual service manual has been developed. The application uses interaction with the Microsoft Kinect device which enables users to control and interact with computer world through a natural user interface using gestures and spoken commands. The device finds greatest application in interactive entertainment, but this paper presents and exploits its great potential in industrial environments.

I. INTRODUCTION

With the rapid development of new technologies we are constantly searching for new methods which would affect our experience of interaction with computer world. *Augmented Reality Technology* (AR) superimposes a computer-generated image on a user's view of the real world, thus providing a composite experience of human-computer interaction. AR techniques are used widely in various spheres of human life, and are especially interesting and attractive to apply in the advertising and interactive entertainment. However, the great potential of AR technology has not yet been sufficiently exploited in modern production processes, which leads us to the question of how to use this technology in order to facilitate work in industrial plants.

AR technology is rather interesting and useful way of exploiting modern computer technology and high performance electronics to blur boundaries of real and virtual world. As explained in [1], AR is a variation of the more known concept of *Virtual Reality Technology* (VR) and it is important to notice the difference between the two. VR technology creates virtual environment in which user feels to be moving inside of a computer-generated world, but at the same time he cannot perceive the real world which still surrounds him. On the contrary, AR allows the user to feel the real world, augmenting it with superimposed virtual objects. Awareness of the real world allows user to act natural in complex environment, but also to feel safe at the same time. Precisely this feature is a great potential for AR application in complex or dangerous maintenance tasks in industrial environments.

As concluded in [2, 3], maintenance tasks in industrial environments are excellent domain for AR applications. However, the great potential has not yet been exploited as well as in the field of entertainment. The basic idea of AR is to bring additional information as seamlessly possible into the view of a user, thus providing new perspective and better understanding of real-world situation. Another advantage of AR systems in industrial environments is fault tolerance which can greatly reduce number of errors during maintenance tasks. Economical point of view is also discussed in [1] where authors agree that industries can use AR to lower processes' operational costs and thus sustain their growth and innovation.

There are many available solutions for AR which can be easily adapted for industrial application. Most AR systems make use of simple handheld devices such smartphones, which are usually equipped with compasses, global position system sensors, gyroscopes and cameras. These kinds of well equipped devices provide a large amount of information about real world which can be easily supplemented with virtual features, thus creating a simple AR system. This paper will address more complex AR system which enables user to interact through *natural user interface* (NUI) using gestures. NUI is alternative to a *command-line interface* (CLI) or *graphical user interface* (GUI), but also their possible successor. It allows user to act and feel more natural in interaction with computer world, and enhances their experience of using computer systems. Motion sensing input devices are usually needed to achieve proper natural interaction, and therefore Microsoft Kinect will also be introduced in the paper.

II. INTERACTIVE MANUALS

Collaboration with Austrian company AVL, which is engaged in development, testing and simulation of powertrain systems with internal combustion engines, has contributed with specific problem which is solved in AR domain. The paper follows the process of developing interactive virtual service manual which facilitates maintenance of *AVL Micro Soot Sensor* (Figure 1).

AVL Micro Soot Sensor is device for continuous measurement of lowest soot concentration in the diluted exhaust from internal combustion engines. Its maintenance presents an extremely demanding and responsible job which is carried out as a series of careful tasks. These tasks are animated using supplied 3D model of the Micro Soot device, and present the main feature of the developed interactive software. Software also enables user to manipulate with the model, e.g. rotate, zoom, disassemble device model and explore all of its parts.

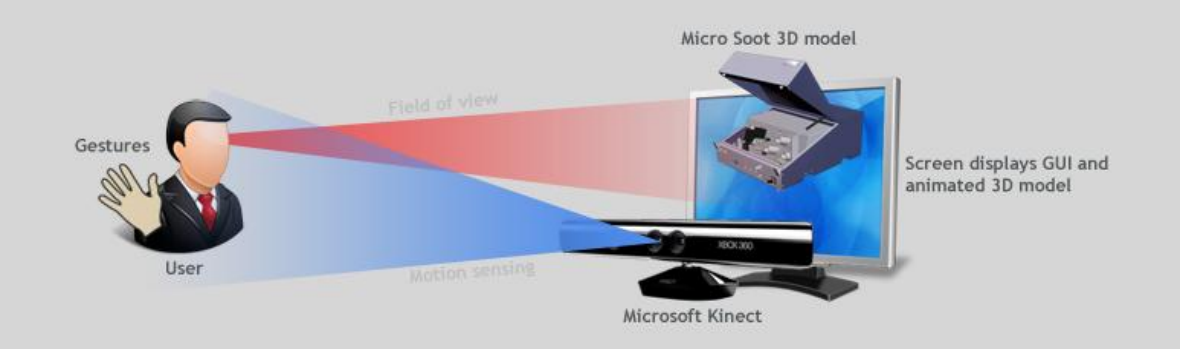

Figure 1 AR system as an interactive virtual service manual

Entire human-computer interaction is implemented using Microsoft Kinect as a motion sensing input device, i.e. user can interact with the device through NUI using gestures. The most of successful Kinect application concern the field of entertainment [1, 4], but this AR system introduces Microsoft Kinect to an entirely new field of application – maintenance in industrial environments.

III. TECHNICAL IMPLEMENTATION DETAILS

During planning and development of this project several problems and lack of features in the standard SDK were encountered, so for this program to function correctly they had to be implemented. Some of the notable examples of these problems and how they were solved will be presented in this segment. The biggest issues were creating an augmented reality environment that uses real 3D graphics engine as a backbone and creating a system that enables intuitive interaction and use of natural user interface.

A. Creating Augmented Reality

To create an augmented reality view where real and virtual world are interconnected, rendering engine and live video feed have to be somehow connected. Since various 3D models (Micro Soot sensor and its parts) and animations (service manual steps) had to be loaded, standard WPF interface couldn't be used because these features aren't supported there so 3D rendering engine has to be used. As it was mentioned before, technologies and SDK*s* that were used are official Microsoft technologies, and were chosen because of compatibility, portability and their ease of use, and for that reason *XNA Game Studio* was chosen as a 3D rendering engine. But creating AR view isn't straightforward. Although Kinect SDK enables the user to get frames from various video feeds (color, depth, infrared) that are retrieved from the Kinect sensor they come in the form of pixel arrays and first have to be converted to a compatible format. For example, bytes from color video frames come in the BGRA format (blue, green, red and alpha channel) and XNA uses RGBA format, so the order of red and blue bytes has to be swapped. When using standard WPF framework color format can be specified as a parameter and conversion takes place automatically but in XNA it has to be done manually and for that purpose small HLSL program was written (since it's a shader everything is done on the GPU so the performance hit is negligible). After the image is converted it has to be drawn to the screen, and to do that first the array of pixels has to be transferred to a 2D texture and then this texture is drawn as a sprite over the full screen. After that, models and GUI (graphical user interface) can also be rendered. Since XNA has an unusual way of deciding what is rendered in which order in the graphical pipeline, special attention has to be given to manual tweaking of rendering order so that the video image is always in the background, models in front of it and GUI on top or problems like drawing invisible or semi-transparent models and GUI may arise.

B. Natural user interface and *Graphical user interface*

Opposed to classical user interfaces where some kind of hardware has to be used (mouse, keyboard, joystick etc.) in combination with various buttons, menus and images to perform a certain action, natural interfaces allow use of voice or intuitive gestures like waving, reaching or grabbing to interact with program with the same purpose like they are used in real world. For example waving might be interpreted as "Hello" or "Goodbye" and could be used to start or stop a program.

In this case grabbing was used as a way to rotate, zoom or disassemble the model. For zooming both hands have to be used and it works in a similar way like a "pinch to zoom" gesture on modern multi-touch smartphones. Problem with using the grabbing gesture is that Kinect doesn't support most gestures "out of the box" so a method to detect whether the fist is open or closed had to be developed [5, 6, 7]. Data that Kinect does provide and which helps in accomplishing this task were skeleton info (contains positions of 20 major joints in the body) and depth image (for each camera pixel its distance from the Kinect is given). First the joint position of each hand is acquired and then using this position as center, rectangular region approximately the size of the open hand is extracted from the depth image. This was done to minimize the amount of scanning (lowers performance costs a lot, instead of checking 640x480 pixels only around 30x30 pixels have to be used). In the next step horizontal and vertical scanning of each line is performed where depth information is used to isolate hand from background objects (no complex computer vision algorithms are needed for edge detection like with conventional 2D cameras). Scanning is used to determine the amount of gaps between fingers (Figure 2) and this information is then used to conclude whether the fist is open or closed (when it is closed there should be no gaps detected). This primitive method works well for a

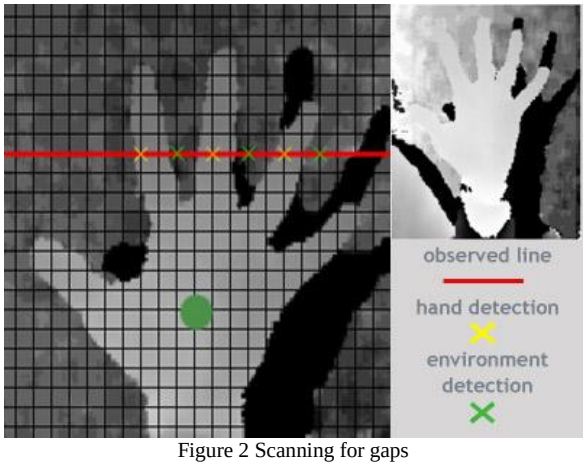

Figure 2 Scanning for gaps

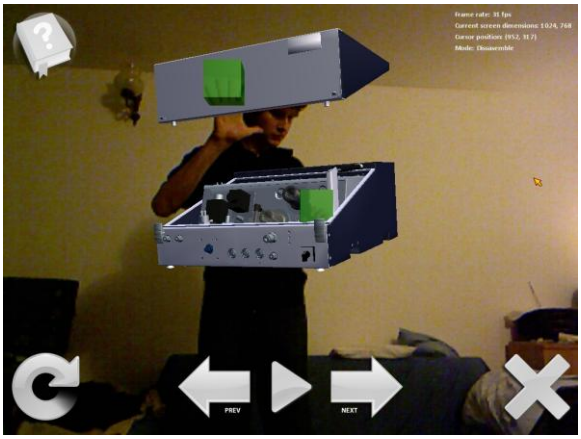
Figure 3 Part moved, hand released

prototype application where false positives can be tolerated but later versions will use some of the more robust algorithms (probably "blob detection" from *OpenCV* library).

After the status of the hand has been detected it can be used for grabbing virtual objects. In the case of model disassembly, grabbing is combined with mesh collision features of the XNA Framework. Usually invisible spheres or boxes are used to approximate the model because of performance reasons. Regarding the box based methods XNA only supports axis aligned bounding boxes and using them would give very bad results when the parts are rotated. Other box based collision method is called object oriented bounding boxes and it is much more precise because it does rotate itself with the model but it is also more computationally expensive and difficult to implement and for our purposes it wouldn't be cost effective to implement it by ourselves. Since the model and parts are rotated a lot, spheres were chosen because they are always the same, regardless of rotation. Minor annoyance is that XNA doesn't automatically calculate the dimensions and positions of the collision spheres depending on the model part but they have to be created manually. Collision detection is activated only if the user has closed his hand and then one sphere is assigned to the position of the hand. The sphere is then checked in a loop with all generated spheres of all parts in the scene. If they intersect then while moving the closed hand grabbed part also moves in the same direction and by the same amount (just like grabbing and moving an object would happen in real world). (Figure 3)

The other two gestures, zoom and rotate, were used in a more 2D way: we take note of the hands vertical and horizontal movement, and ignore the forward/backward movement. The reasoning behind this reduction in dimensions is user oriented – the user is viewing his actions on a 2D surface, it's not intuitive and it's hard to see your hands position in 3D relation to the virtual model. Because of tracking errors we had to include a threshold to our movement detection, and we also had to use fixed values for frame-to-frame scale and rotation updates as to avoid jagged movements of the model. One more point that distinguishes tracking hands with Kinect from tracking with a webcam is that Kinect preserves spatial distances. For example, Kinect will detect that you moved your hands 10cm apart if you're standing 2, 3 or 5m away from the device, while a webcam would detect different distances depending how far away you are form the camera.

Since the program has rich functionality not every action can be performed by gestures. Some of the actions wouldn't be intuitive and would be easily forgettable. this way the meaning of using NUI for its simplicity would be lost, so a classic GUI also had to be created, but it can be used with Kinect and with a mouse. Selection of buttons is performed by placing the hand or mouse on top of it and actions are confirmed, with the mouse by clicking, or hovering for a second over the button with a hand (Figure 4). Hovering was chosen because it is one of the methods described in [8] that can be used with classic GUI. It is also stated that a clear visual feedback of what is happening has to be provided. Therefore the buttons get highlighted when they are selected and while activating the button, the status indicator for the hand is getting more transparent while time passes, which indicates that something is happening.

C. Animations

In order to provide illustrative maintenance instructions, each task was animated in Autodesk 3D Studio Max, and then loaded into XNA program. This way the user is enabled to walk through animated service steps using intuitive navigation buttons.

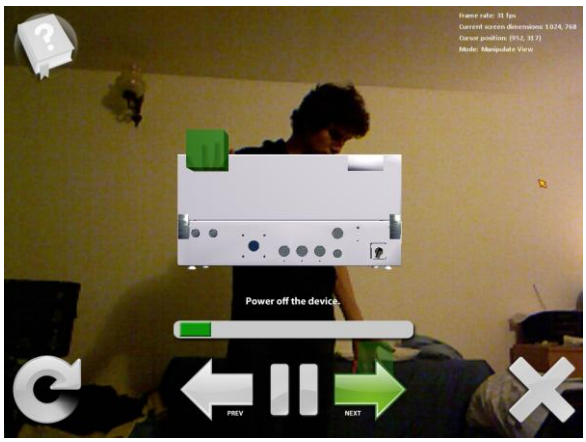

Figure 4 Button selection and activation

Seven animations were created and exported into separate FBX files. These files contain only animations and no geometry. Model meshes were culled down to one triangle reducing the size of the files. These "naked" files are loaded into program where transform information is extracted from them and applied to a preloaded unanimated model with complete meshes, i.e. reference model. The principle is described by Dr. Charles Owen in [9]. Owen developed this solution for loading pre-animated models into XNA programs. The solution extends *Content Pipeline*, which is responsible for loading assets into an object that can be used in XNA code. A custom made content processor class processes imported models and extracts animation data. Special animation data classes were used by the animated model processor to store the animation data and by the software application to load this data at runtime [10].

IV. CONLUSION

The most successful AR applications concern the field of entertainment. However, its ability to provide new perspective and better understanding of real-world situations guarantees wider future application in any kind of industry. Particularly promising areas of application are maintenance tasks in industrial environments. AR technology has proved to be highly suitable for development of interactive service manuals which can greatly facilitate complex maintenance tasks.

Introducing Microsoft Kinect to a new field of application has provided new exciting features in interactive manuals such as NUI. NUI implementation has proved to be very attractive, highly useful and profitable. It allows the user to act and feel more natural in interacting with computer world, and enhances his experience of using interactive manuals. The user can explore task problems more deeply, while feeling extra safe in a fault-tolerant environment. Both users' hands are free while interacting with manual which is sometimes of utmost importance in harsh environments of industrial plants. In the near future NUI will most certainly find greater use in industrial environments, while today its potential is still limited to the field of entertainment.

ACKNOWLEDGMENT

Special thanks are due to all partners from AVL Company, especially to Croatian representatives in Zagreb.

REFERENCES

[1] M. Hincapie, A. Caponio, H. Rios and E.G. Mendivil, "An introduction to Augmented Reality with applications in aeronautical maintenance," *Transparent Optical Networks (ICTON), 2011 13th International Conference on* , vol., no., pp.1-4, 26-30 June 2011.

[2] S. Bernd and L. De Blandine, "An augmented reality system for training and assistance to maintenance in the industrial context," *Journal of Winter School of Computer Graphics*, vol. 11, pp. 101-110, 2003.

[3] N. Zenati, N. Zerhouni and K. Achour, "Assistance to maintenance in industrial process using an augmented reality system," *Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on* , vol.2, no., pp. 848- 852 Vol. 2, 8-10 Dec. 2004.

[4] T. Leyvand, C. Meekhof, Yi-Chen Wei, Jian Sun and Baining Guo, "Kinect Identity: Technology and Experience," *Computer* , vol.44, no.4, pp.94-96, April 2011.

[5] M. Tang, "Recognizing Hand Gestures with Microsoft's Kinect," Stanford University, Department of Electrical Engineering, Technical Report, March 16, 2011., unpublished

[6] A. Drake, "Kinect Hand Recognition and Tracking," Washington University in St. Louis, Kinect Hand Recognition and Tracking, Project Report, May 1, 2012., unpublished

[7] F.T. Cerezo, "3D Hand and Finger Recognition using Kinect," University of Granada, Project Report, 2011., unpublished

[8] *Human Interface Guidelines for Kinect for Windows*, Vers. 1.5.0., 2012.

[9] C. Owen, "A Better XNA Skinned Sample," *http://metlab.cse.msu.edu/betterskinned.html*, October 24, 2012.

[10] A. S. Lobao, B. Evangelista, J. A. Leal de Farias and R. Grootjans, *Beginning XNA 3.0 Game Programming: From Novice to Professional*, New York: Springer-Verlag, 2009