# An Advanced Authoring Tool for Augmented Reality Applications in Industry

Jesús Gimeno, Pedro Morillo, Juan M. Orduña, Marcos Fernández [1]

*Abstract*— **The use of authoring tools has become a valuable trend for the fast development of Augmented Reality (AR) applications in industrial organizations. However, most of current AR authoring tools are actually programming interfaces that are exclusively suitable for programmers, and they do not provide advanced visual effects. In this paper, we propose an easy-to-use AR authoring tool oriented to the development of AR applications for the execution of industrial sequential procedures. Unlike other recent easy-to-use AR authoring tools, this software framework allows non-programming users to develop low-cost AR applications, including occlusion capabilities, by means of the use of a Kinect sensor. The evaluation results show that overlaying 3D instructions on the actual work pieces reduces the error rate for an assembly task by more than a 75%, particularly diminishing cumulative errors common in sequential procedures. Also, the results show that the time required by non-programming users to develop the AR prototypes using our tool was more than 90% lower than the time required for developing the same prototypes with computer graphics programmers.**

*Key words*— **Augmented Reality, Authoring Tools, Non-immersive Desktop, Kinect, Occlusion**

## I. Introduction

AU gmented Reality (AR) systems have been widely used in numerous applications such as medical procedures, scientific visualization, manufacturing automation, cultural heritage and military applications [1].

One of the main problems that prevents AR applications to become popular is the lack of AR authoring platforms that allow unqualified users in computer graphics to easily generate AR applications. There are popular software libraries like ARToolKit [2] and ARToolKitPlus [3] that use OpenGL, VRML or OpenSceneGraph [4] to represent the 3D models on the real images in real time. However, the use of these and others computer graphics libraries requires programming skills to generate AR applications, and every AR development should be constructed from the scratch.

In order to avoid these problems, AR authoring tools were proposed a decade ago [5]. Later, an extensible authoring tool that supports both scripting and a drag and drop interface and real time interpreted input was developed [6]. A recent work even classifies the existing AR tools depending on the use of external libraries, and the programming knowledge required for using them [7].

Assembly, maintenance and even repair tasks are some of the direct application fields of AR tools, and a lot of proposals have been made in these industrial areas. A recent work [7] classifies AR development tools into two categories: AR-related software framework and GUI-based AR authoring tools.Nevertheless, once a mechanic begins to physically manipulating objects in an task, he does not always need the visual information provided by the display [8] to complete certain steps of a given industrial procedure. On other hand, although several AR systems have been proposed for industrial purposes, most of them superimpose the computer-generated objects on the real view of qualified workers. This forced superposition cause the occlusion problem, which occurs in AR systems when a computer-generated object closer to the viewer obscures the view of real elements farther away along the line-of-sight [9]. If the occlusion problem is not properly addressed in the development of an AR system for industrial purposes, then the developed tool does not significantly facilitate workers their actual on-the-job tasks [10].

Figure 1 shows an example of the occlusion problem when a custom AR system has been used in the on-site repair process of a six-cylinder diesel engine. Concretely, the pictures included in this figure show the step when the first of six fuel injectors (modeled using eye-catching green colors) is fitted into the proper engine cylinder head. The top picture of this figure shows how a non-ocludded 3D computer-generated fuel injector is visualized over the engine indicating the mechanic a misleading final location of the cylinder head. On the contrary, the bottom picture of the same figure shows how this augmented fuel injector has been correctly occluded by the foreground real objects in the scene, indicating the proper location of the cylinder head within the back side of the engine.

In this paper, we propose an easy-to-use AR authoring tool including occlusion capabilities. This AR tool, called SUGAR, has been intentionally designed to enable a rapid prototyping of low-cost AR systems. Unlike other authoring tools, our approach uses Kinect [11] for computing a depth map of the scene to produce occlusion effects. The application examples show that, unlike other recent easy-to-use AR authoring tools [7], the proposed tool can be used as a general-purpose and low-cost framework for generating different maintenance and repair AR applications. Also, this paper describes some experiments that test the relative effectiveness of AR instructions in assembly, maintenance and repair tasks. The evaluation results show that overlaying 3D instructions on the actual work pieces reduced the error rate for an assembly task by 79%, particularly

[1]Departamento de Informática, Universidad de Valencia, e-mail: {Jesus.Gimeno, Pedro.Morillo, Juan.Orduna, Marcos.Fernandez}@uv.es

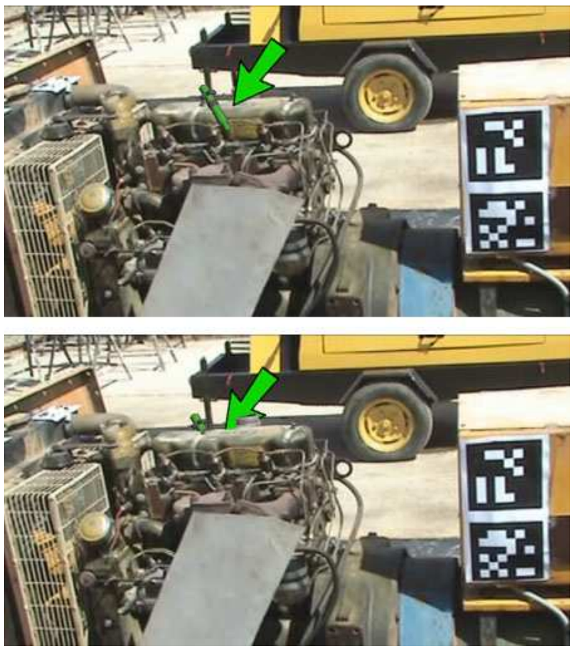
Fig. 1. An example of the occlusion problem in an AR system for industrial purposes

diminishing cumulative errors (errors due to previous assembly mistakes). Moreover, the time required by non-programming users using SUGAR to develop the AR prototypes was much lower (approximately a 95%) than the time required for developing the same prototypes with expert programmers. These results validates the proposed tool as a general-purpose AR authoring tool for industrial AR applications.

The rest of the paper is organized as follows: Section II describes in detail the proposed AR authoring tool. Next, Section III shows different application examples of the proposed tool, and Section IV shows the performance evaluation of AR instructions in an assembly task using the proposed tool. Finally, section V shows some concluding remarks and the future work to be done.

## II. An overview of SUGAR

SUGAR (which stands for **S**ystem for the development of **U**nexpensive and **G**raphical **A**ugmented **R**eality application) is an open-source software platform designed to enable a rapid prototyping of low-cost AR systems. Our framework is oriented to develop complex AR software applications based on procedural simulations, which are modeled following an easy-to-use AR authoring editor. This AR editor generates an exchange file, describing the AR procedure, which can be loaded into different AR devices not requiring high computational power.

SUGAR allows the creation of virtual contents through an easy edition mechanism based on real world photos. The user takes photos of the real scenarios to be augmented, and is on these photos where the virtual content is edited. The ARToolKitPlus-based markers are automatically generated with a simple calibration step, in such a way that the user should only paste the AR markers on the real object in order to visualize the augmented scene. Therefore, SUGAR bridges the gap between virtual information and the real world, offering the user an easy way of creating virtual contents according to reality.

The SUGAR modules can be classified in two groups: description of the real world, and virtual content edition. The first group includes those modules necessary for creating the scenarios where the virtual contents will be displayed. Each scenario is composed of an image of the real environment, some AR markers that are generated automatically, and a depth map (the latter one only is presented if a Kinect is available when the photo is taken). In order to create this scenario, two wizards guide the user through some easy stages. The first stage, called the *PhotoKinect Wizard module*, displays the real-time image of the markers camera and allows taking photos storing at the same time the depth image. This depth information will be used later in order to produce correct occlusions. The other wizard, called *Locations wizard*, allows the user the creation of a scenario from either a conventional photo or a photo captured with AR markers. This wizard is also split into four easy substages: first, the desired images are loaded. Then, the wizard allows to draw a rectangle indicating a flat area where the markers can be located. The two final substages consists of the introduction of the flat area defined within the real image and the number of AR markers to be used. After these steps, the scenario is ready for editing the virtual contents. The virtual models will be adjusted to the size of the real object to be augmented by using the size information introduced by the user. At any moment, the user can select a scenario and print the associated markers in order to paste these markers on a real object. This simple wizard allows a user with neither programming, nor ARToolKit/ARToolKitPlus knowledge, the creation of an AR marker with the proper size and the corresponding associated configuration file.

After the creation of the scenarios, the other group of modules includes the procedures for defining the virtual information. The editor of SUGAR uses a structure based on steps (denoted also as slides), where each of them is associated to one of the scenarios previously created. The edition of the virtual content can be split into three different stages: definition of the current step of the procedure, creation of the virtual content associated to each slide, and the definition of the corresponding evaluation tests for this step, if necessary. The definition of the slides includes some basic office functions: creation, ordering, text edition, aspect, associated video, associated pdf files, etc. All these functions can be easily performed, in a similar way to the creation of a conventional graphic presentation.

The creation of virtual content consists of adding some virtual elements that help to explain each step to the scenario displayed in the slide. The virtual elements can be created either from basic 3D objects organized in an internal 3D library, or loading 3D models previously created using Autodesk 3DS Max. The included 3D library consists of cubes, spheres, planes, cones, etc., which can be grouped to generate more complex models, and allows changing their textures, colors, or other properties. Also, we have developed an animation module based on keyframes that allows to animate the virtual objects. In order to allow the creation of 3D models that are appropriate for the scenario to be augmented, the 3D scene includes a template with the photo that is used as the model, and even (depending on the type of scenery) a mesh created from the picture taken with Kinect. Using this reference, the user only has to locate the virtual models on the template. This way of creating a scene, starting from a reference image, allows the location of the virtual elements and their size adjustment.

## A. *Software Architecture*

The software architecture of SUGAR is based on a modular approach oriented to develop procedural AR systems. Basically, the SUGAR framework consists of two applications: an easy-to-use editor for AR procedures and an AR light viewer. Both applications share some software modules permitting them to reuse 3D-tracking data structure models, and visualization code. Figure 2 shows the software architecture of the SUGAR framework for the development of AR environments. Both applications ("SUGAR Editor" and "SUGAR Viewer") share the kernel of the AR framework denoted as "Data Core". This kernel provides basic services for augmented reality facilities as camera tracking, marker handling and virtual object interaction.

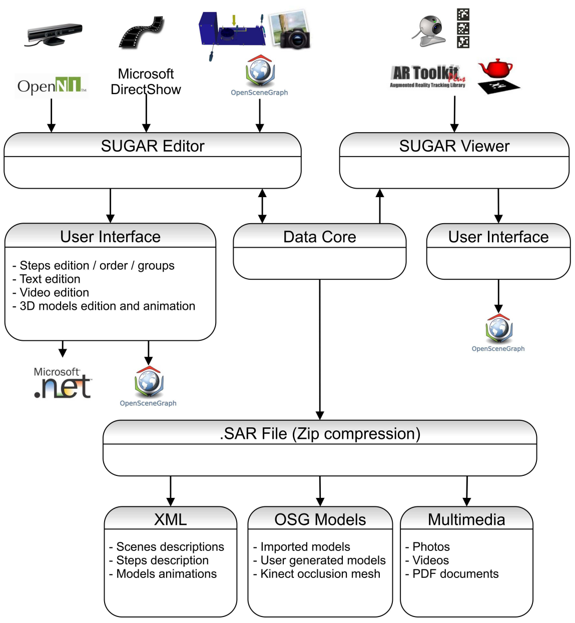

Fig. 2. A modular view of the software architecture in SUGAR

Although AR editor and AR viewer share this software module, each application includes a different user's interface. Thus, the graphical user interface of the AR editor has been developed on Windows Forms and includes some components, developed on OpenSceneGraph, in order to offer high-performance 3D graphical capabilities when users create and edit the AR procedures. In this sense, the OpenNI software library allows accessing the depth map of the real scene using the Kinect device. Moreover, the Microsoft DirectShow API enables high-quality playback of streaming video and audio to be attached to the steps of the industrial (maintenance, repair or even assembly) procedures. In contraposition to this multiview 3D application, the AR viewer corresponds to a light Windows application, developed on Qt/C++ [12], embedding a reduced version of the OpenSceneGraph framework and including a reduced set of primitives for this 3D high-level library tool. Since Qt/C++ is a cross-platform development framework, the same source code for a Qt application can be compiled on different operating systems without changes. In this sense, the "SUGAR Viewer" has been ported to Symbian and Android operative systems with minimum changes within the initial source code. The reduced computation workload of this multiplatform application allows the viewer to be executed on low-power general purpose processors to execute the AR procedures in on-site industrial environments.

The "Data Core" module also includes the definition and the basic properties of the exchange file format for the SUGAR framework, denoted as SAR files. These files are generated by the AR editor to be imported by the SUGAR multiplatform viewers. Basically, the SAR files are zip-compressed archives containing a structured representation of AR applications, and the corresponding multimedia content, that can be attached to the steps of the modeled AR procedures for industrial task. The organization of the digital content, including the depth maps obtained from the Kinect device, has been defined using XML files, providing a flexible configuration mechanism.

## III. Application Examples

In order to validate our AR editor as an efficient tool for the rapid prototyping of AR applications for assembly, maintenance and repair purposes, we have tested our tool in four different application examples belonging to various industrial areas. Concretely, these application examples are the following procedures: the replacement of the cut heading within a lathe machine (metal machining area), the assembly of a computer starting from its basic components (computer and electronics manufacturing area), the repair of the admission system in a mobile lighting tower (maintenance of heavy machinery area), and the review of the spark plugs and the ignition coils on a BMW M3 E92 (420CV) engine (automobile maintenance area). We have denoted these procedures as *PROC 1* to *PROC 4*, respectively. Table I shows the amount of steps, gathered into groups, needed to complete each one of the tasks. Since not all the steps recommended by the manufacturer need visual indications to be completed, this table also shows, for each procedure, the number of steps that actually include AR contents.

TABLE I
Decomposition of the procedures in steps

| Procedure | Groups | Steps | AR Steps |
|---|---|---|---|
| PROC 1 | 6 | 51 | 26 |
| PROC 2 | 5 | 25 | 20 |
| PROC 3 | 7 | 58 | 32 |
| PROC 4 | 4 | 15 | 10 |

In order to help in the completion of these tasks, SUGAR authoring tool was used to prototype two different AR systems (for each of the four procedures considered): a computer assisted instruction system using a TabletPC system (we will denote this AR system as S2) and a computer assisted instruction system using a head-mounted display (we will denote this AR system as S3). Concretely, the S2 system was developed on a Dell Latitude XT1 TabletPc (Intel Core 2 Duo at 1.2GHz, 2GB RAM, ATI Radeon Xpress 1250, Windows 7 Professional) including a LCD-CCFL 12.1-inch screen for outdoor environments, reaching up to 400cd/m2 of brightness. The S3 system consists of the same Dell Latitude XT1 but connected to an "AR Vision 3D HMD", which is a full stereoscopic video see-through AR device including a binocular SVGA (1024x768) displays and two separate 752x480 (60 FPS) color cameras. In order to illustrate the considered application examples, Figure 3 shows a snapshot of the proposed AR editor (SUGAR Editor) when implementing Procedure 1.

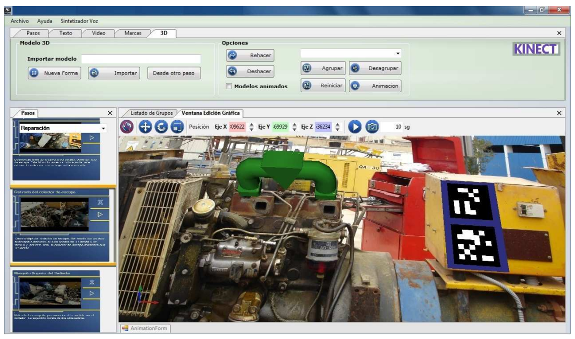

Fig. 3. SUGAR Snapshot for Procedure 1

Figure 3 shows the interface of the proposed AR editor for sequential procedures included in SUGAR. The easy-to-use interface of the editor is very similar to the most common graphic presentation programs (such as Microsoft PowerPoint, Apple KeyNote, Open Office Impress, etc.), and it consists of a large view of the current step of the modeled procedure along with a view of the rest of steps (as a slide sorter) on the left side of the screen. The editor allows non-programming users to create AR applications consisting of a set of sequential steps handled by AR markers. In this sense, the toolbars on the top of the window allow to include multimedia contents such as video, images, text, manuals and AR information into the steps of the AR procedure. Figure 4 illustrates the use of the S2 system when Procedure 1 was tested in a real industrial environment.

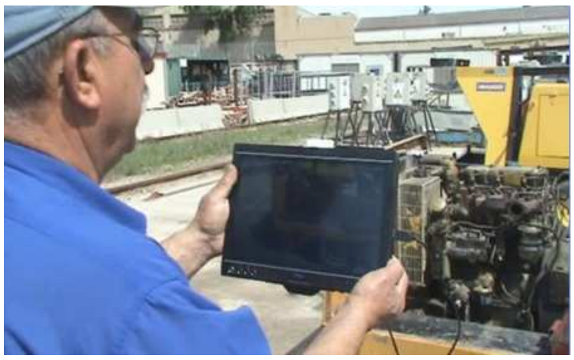

Fig. 4. Real use of S2 system in Procedure 1

## IV. Performance Evaluation

The performance evaluation of augmented reality authoring tools results in a complex task, since it is very difficult to quantitatively show the utility or effectiveness of the proposed methodology and tools. The measurement of costs reduction in software development and maintenance neither results an easy task. The main reasons are the lack of objective metrics in not only Augmented Reality, but also Virtual Reality, and the qualitative and even fuzzy nature of most software engineering studies [13].

A possible way of evaluating the performance of Augmented Reality tools is a qualitative, user-centered approach [14]. According to recent studies [15], the observations and questionnaire are the basis for a qualitative analysis. The questionnaire consists of several questions where the participants in the evaluation of the AR tool can freely answer on their experience of the AR system. The questions are usually related to overall impression of the AR system, experienced difficulties, experienced positive aspects, what they would change in the system and whether it is possible to compare receiving AR instructions to receiving instructions from an instructor. However, a qualitative approach does not allow neither to compare different AR tools on a fair basis, nor to evaluate their performance in comparison with traditional training or guidance tools. On the contrary, a quantitative analysis can allow a fair comparison of different tools. Some metrics like cost efficiency, development time and maintainability have been proposed for a quantitative analysis [16]. However, that work does not define concrete criteria for assigning values to the three metrics, and only the development time is accurately measured.

We propose a quantitative approach for the performance evaluation of the AR tool in some industrial environments. In order to measure the complexity of the assembly, repair and maintenance tasks, we have followed the criteria proposed in [17]. Therefore, we have classified Procedure 1 as very complex, Procedure 2 as normal, and Procedures 3 and 4 as complex.

In order to measure the performance provided by the AR systems prototyped with SUGAR, we have first measured the average completion time required by fifteen different users in order to completely execute each of the considered procedures. All the users were experts technicians in their area, but a training session was performed prior to the performance evaluation in order to allow the users to get in contact with the AR technology. The same users were used for performing the four procedures, in order to avoid any skew in the results due to the different skill of different populations. However, the fifteen users were divided into four groups, and the sequence of procedures executed by each group was different, in order to avoid skews in the experimental results due to the potential training with the technology. Moreover, not only it was the first time that the users executed the considered procedures, but also any of the groups repeated the same procedure using different systems, in order to avoid the benefit with the knowledge and experience that they acquire before.

For comparison purposes, the users also performed the considered procedures exclusively using a printed manual provided by the manufacturer. We have denoted this ”system” as S1. Table II shows the average completion times required when using each system for all the procedures considered. As it could be expected, the average completion times for systems S2 and S3 are much lower than the ones achieved with S1 system. Also, this table shows that the times achieved with S3 (computer assisted instructions (CAI) using a head-mounted display) are lower than the ones achieved with S2 (CAI using a TabletPC display), reaching even less than half the time required for the same procedure with system S1 (in Procedures 1, 3 and 4). These results show the significant benefits that AR systems can provide to assembly, repair and maintenance tasks.

TABLE II
Completion times with different systems

| Procedure | S1 | S2 | S3 |
|---|---|---|---|
| PROC 1 | 4h 30min | 2h 30min | 1h 45min |
| PROC 2 | 50min | 35min | 30min |
| PROC 3 | 6 h | 3h 30min | 2h 45 min |
| PROC 4 | 2h | 1h 15min | 50min |

In order to get a more in-depth analysis of the results shown in Table II, we have measured the mistakes made when executing the AR procedures for all the considered experiments. Table III shows the average number of steps in the experiments that were repeated by the qualified participants because of human errors. As it could be expected, the direct visual guidance provided by the augmented reality devices in systems S2 and S3 resulted in a significant decrease in the number of repeated stages, compared to the use of manufacture's manuals in S1. The system S2 provides the lowest number of repeated stages. Since the execution of assembly and maintenance/repair tasks are often incremental processes, where the results of previous steps are the input of next steps, undetected mistakes in preceding stages (denoted as cumulative errors) result in repeating all the previous affected tasks, starting where the mistake was committed. For this reason, the number of the repeated tasks grows exponentially as the errors made by the participants increase. Nevertheless, the differences between S2 and S3 systems become significant even if taking into account this fact, showing that immersive augmented reality does not provide the best performance in industrial environments.

TABLE III
Number of repeated stages when completing the ARprocedures

| Procedure | S1 | S2 | S3 |
|---|---|---|---|
| PROC 1 | 12.25 | 2.65 | 3.30 |
| PROC 2 | 7.51 | 1.26 | 4.65 |
| PROC 3 | 14.13 | 2.40 | 3.60 |
| PROC 4 | 4.33 | 1.23 | 1.57 |

Nevertheless, Tables II and III do not actually measure the performance of SUGAR, but the prototypes developed with this AR authoring tool. In order to measure the performance achieved with SUGAR, we have asked two different teams to develop the prototypes whose results are shown in Table II. One of the teams (we will denote this one as Team 1) was composed of AR programmers, and the other one (we will denote this one as Team 2) was exclusively composed of expert technicians. As a reference, we asked Team 1 to develop the prototype following the classic AR development approach, by writing, compiling and debugging source code. In this sense, Team 1 developed the AR prototypes using Microsoft Visual Studio 2010 as C++ development framework, OpenSceneGraph 2.9.7 as 3D graphics visualization toolkit and ARToolKitPlus 2.2 as software library for AR purposes. In order to measure the performance of SUGAR, we asked Team 2 to develop the same prototypes with SUGAR.

Table IV shows the development times (in working days) required by each team. The column labeled as “Team” shows the team, the column labeled as “Param.” shows the specific parameter of the development time measured by each row, and the other four columns shows the results for each procedure. The parameters measured are the following ones: SLOC measures the final number of source lines of code included within the final AR prototype, while FPS indicates the frame-rate (in frames per second) achieved using the test hardware. The parameter CT measures the number of working days required by the team for completing the coding stage of the prototype. The parameter DT measures the number of working days required by the team for completing the debugging/adjusting stage, and the parameter TT measures the total time required for the development of the prototype (the sum of CT and DT parameters).

Table IV shows that the size of the source code generated by SUGAR is roughly a 10% higher than the source code created by traditional AR programming. However, this slight increase of the source code

TABLE IV

Source-code sizes and development times

| Param. | PROC 1 | PROC 2 | PROC 3 | PROC 4 |
|---|---|---|---|---|
| | | Team 1 | | |
| SLOC | 71853 | 64710 | 76254 | 53695 |
| FPS | 28 | 44 | 32 | 50 |
| CT | 95 | 76 | 108 | 59 |
| DT | 15 | 10 | 17 | 6 |
| TT | 110 | 86 | 125 | 65 |
| | | Team 2 | | |
| SLOC | 79215 | 71523 | 37749 | 60101 |
| FPS | 31 | 40 | 35 | 50 |
| CT | 3 | 3 | 2 | 2 |
| DT | 1 | 1 | 2 | 1 |
| TT | 4 | 4 | 4 | 3 |

does not have an effect on the graphic performance of the AR application. Moreover, Table IV also shows that the time required by non-programming users with SUGAR to develop the AR prototypes are less than 5% of the ones required for developing the same prototypes with programmers. These differences of orders of magnitude show the potential that an intuitive AR authoring tool like SUGAR can provide to industrial environments. Moreover, the ease of use of SUGAR allows to avoid the need for programming skills, exclusively requiring the expertise of technicians in that field for developing AR prototypes. Also, any potential change required by the prototype can also be made by technicians, without the need of programming skills.

## V. Conclusions

In this paper, we have proposed an easy-to-use AR authoring tool with occlusion capabilities which allows the easy creation of interactive augmented reality applications without any programming knowledge. Unlike other recent proposals, our tool does not rely on expensive or unavailable 3D models, and it uses Kinect for computing a depth map of the scene.

The performance evaluation results show that the registration of geometric models to the real-world counterparts in industrial procedures significantly facilitate workers their actual on-the-job tasks. The direct visual guidance provided by the AR devices significantly decrease the number of repeated stages when compared to commonly used manufacture's manuals. Also, the time required by users with non-programming skills to develop the AR prototypes using our tool was much lower than the time required for developing the same prototypes with expert programmers when following a classical development of AR systems. These results shows the potential of our approach and validates it as a general-purpose AR authoring tool for industrial AR applications.

## Acknowledgements

This work has been jointly supported by the Spanish MICINN and the European Commission FEDER funds under grants Consolider-Ingenio CSD2006-00046 and TIN2009-14475-C04-04.

## References

[1] S. Cawood and M. Fiala, *Augmented Reality: A Practical Guide*, Pragmatic Bookshelf, 2008.

[2] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings of International Workshop on Augmented Reality (IWAR'99)*, 1999, pp. 85–94.

[3] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," in *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, 2007, pp. 139–146.

[4] D. Burns and R. Osfield, "Open scene graph a: Introduction, b: Examples and applications," in *Proceedings of the IEEE Virtual Reality Conference 2004 (VR 2004)*, 2004, p. 265.

[5] Matthias Haringer and Holger T. Regenbrecht, "A pragmatic approach to augmented reality authoring," in *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2002, ISMAR'02, pp. 237–, IEEE Computer Society.

[6] H. Seichter, J. Looser, and M. Billinghurst, "Composar: An intuitive tool for authoring ar applications," in *In Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'08)*, 2008, pp. 177–178.

[7] M.J. Wang, C.H. Tseng, and C.Y Shen, "An easy to use augmented reality authoring tool for use in examination purpose," *Human-Computer Interaction*, vol. 332, pp. 285–288, 2010.

[8] Steven J. Henderson and Steven Feiner, "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*. 2009, pp. 135–144, IEEE Computer Society.

[9] David E. Breen, Ross T. Whitaker, Eric Rose, and Mihran Tuceryan, "Interactive occlusion and automatic object placement for augmented reality," *Computer Graphics Forum*, vol. 15, no. 3, pp. 11–22, 1996.

[10] S. Bong-Kee Sinb P. Sang-Cheol, L. Sung-Hoon and L. Seong-Whan, "Tracking non-rigid objects using probabilistic hausdorff distance matching," *Pattern Recognition*, vol. 38, no. 12, pp. 2373–2384, 2005.

[11] Eduardo Souza Santos, Edgard A. Lamounier, and Alexandre Cardoso, "Interaction in augmented reality environments using kinect," in *Proceedings of the 2011 XIII Symposium on Virtual Reality*, Washington, DC, USA, 2011, SVR '11, pp. 112–121, IEEE Computer Society.

[12] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 4*, Open Source Software Development Series. Prentice Hall, 2008.

[13] J. Seo and S. OhM, "Pvot: An interactive authoring tool for virtual reality," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 7, no. 4, pp. 17–26, 2007.

[14] M. Traskback, T. Koskinen, and M. Nieminenl, "User-centred evaluation criteria for a mixed reality authoring application," in *Proc. of Tenth International Conference on Human-Computer Interaction (HCI)*, 2003, pp. 1263–1267.

[15] S. Nilsson, B. Johansson, and A. JÃ¶nsson, *The Engineering of Mixed Reality Systems*, chapter A Holistic Approach to Design and Evaluation of Mixed Reality Systems, pp. 33–55, Human-Computer Interaction Series. Springer, 2010.

[16] Daniel F. Abawi, Jose Luis, Los Arcos, Michael Haller, Werner Hartmann, Kalle Huhtala, and Marjaana Traskback, "A mixed reality museum guide: The challenges and its realization," in *Proceedings of the 10th International Conference on Virtual Systems and Multimedia (VSMM 2004)*, 2004.

[17] Donald J Campbell, "Task complexity: A review and analysis," *Academy of Management Review*, vol. 13, no. 1, pp. 40, 1988.