Computer-Aided Civil and Infrastructure Engineering **25** (2010) 3–19

Scalable Algorithm for Resolving Incorrect Occlusion in Dynamic Augmented Reality Engineering Environments

Amir H. Behzadan

Department of Construction Management and Civil Engineering Technology, The City University of New York, New York City College of Technology, Brooklyn, NY, USA

&

Vineet R. Kamat*

Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, USA

Abstract: *Augmented reality (AR) offers significant potential in construction, manufacturing, and other engineering disciplines that employ graphical visualization to plan and design their operations. As a result of introducing real-world objects into the visualization, less virtual models have to be deployed to create a realistic visual output that directly translates into less time and effort required to create, render, manipulate, manage, and update three-dimensional (3D) virtual contents (CAD model engineering) of the animated scene. At the same time, using the existing layout of land or plant as the background of visualization significantly alleviates the need to collect data about the surrounding environment prior to creating the final visualization while providing visually convincing representations of the processes being studied. In an AR animation, virtual and real objects must be simultaneously managed and accurately displayed to a user to create a visually convincing illusion of their coexistence and interaction. A critical challenge impeding this objective is the problem of incorrect occlusion that manifests itself when real objects in an AR scene partially or wholly block the view of virtual objects. In the presented research, a new AR occlusion handling system based on depth-sensing algorithms and frame buffer manipulation techniques was designed and implemented. This algorithm is capable of resolving incorrect occlusion occurring in dynamic AR environments in real time using depth-sensing equipment such as laser detection and ranging (LADAR) devices, and can be integrated into any mobile AR platform that allows a user to navigate freely and observe a dynamic AR scene from any vantage position.*

*To whom correspondence should be addressed. E-mail: *vkamat@umich.edu.*

DOI: 10.1111/j.1467-8667.2009.00601.x

1 INTRODUCTION

Three-dimensional (3D) visualization of engineering operations has gained widespread applicability over the past few years. Several researchers have investigated the application of virtual reality (VR) to animate simulated construction operations to verify and validate the results of underlying simulation models (Op den Bosch, 1994; Barnes, 1997; Bishop and Balci, 1990; Rohrer, 2000; Rohrer and McGregor, 2002; Kamat, 2003). To create realistic VR displays of a simulated process, detailed data about the process as well as the environment

in which it takes place have to be obtained. Such data must be able to describe the simulation, 3D Computer-Aided Design (CAD) models and their interactions, facility under construction, and terrain topography. As the size and complexity of the operation increases, data collection also becomes time and resource consuming. This directly translates into loss of project financial and human resources that could otherwise be saved and used more productively.

Augmented reality (AR), on the other hand, is a fast-emerging technology with a great potential in visualizing and communicating results of engineering and scientific simulation models at operations level of detail (Webster et al., 1996; Thomas et al., 2000; Gleue and Dähne, 2001; Livingston et al., 2002; Roberts et al., 2002). The application of AR can significantly reduce the time and effort required for CAD model engineering (Brooks, 1999), and at the same time can increase credibility through visually convincing graphical representations of the operations being studied. As a trade-off, however, an AR animation must be capable of managing the visual interaction between two distinct groups of objects: virtual and real. This is critical in AR as the observer of an animation expects to see a mixed scene of seamlessly merged real and virtual objects in which both groups appear to coexist in a realistic manner. This introduces a number of challenges unique to creating AR animations. One of these challenges is the interaction between real and virtual objects. In a dynamic AR environment, interaction can be grouped into two main categories: visual and spatial (Breen et al., 1995). Visual interaction between real and virtual objects is the result of reflection, absorption, and redirection of light emitted from and incident on the objects. Such effects include shadows, reflections, refraction, color bleeding, and occlusion (Shreiner et al., 2004). Spatial interaction between real and virtual objects include kinematic constraints, collision, and response to external forces (e.g., deflection and bending) (Hegror et al., 1989). Kinematic interaction involves constraints or effects created by the motion of one object (real or virtual) on the position and orientation of another object. In collision detection, a number of calculations are performed to determine when and where an object strikes another, thus preventing them from occupying the same physical space. The most complex type of spatial interaction, however, occurs when there is an exchange of force and momentum between real and virtual objects. The studied interaction in this category has typically been one way (i.e., real objects can affect virtual objects, but virtual objects cannot affect real ones) (Noma et al., 1996; Dubois and Nigay, 2000).

The authors have successfully designed and implemented an AR-based mobile visualization system called ARVISCOPE (Behzadan, 2008) that enables creating dynamic animations of simulated operations in real time by combining views of real facilities on the jobsite and virtual CAD objects under construction. ARVISCOPE takes advantage of global positioning system (GPS) positional and head orientation tracking data to place virtual objects relative to the observer in the scene and constantly update their position and orientation (Behzadan and Kamat, 2007). A major part of this research, which is the primary focus of this article, has been to address the problem of visual occlusion between real and virtual objects in an augmented scene. Incorrect visual occlusion can occur when a real object partially or wholly blocks the observer's view of a virtual object. In a dynamic scene such as a construction operation, incorrect occlusion can manifest very frequently and unexpectedly because the real and virtual resources and personnel can move arbitrarily with no constraints. Figure 1 shows a snapshot of an AR animation in which a virtual CAD model of an excavator is superimposed on the real scenes of the jobsite. In this figure, two real objects (i.e., light pole and the real excavator) are closer to the viewpoint and hence must partially block the virtual excavator at two locations (i.e., stick and bucket). However, the observer of the scene views the snapshot in Figure 1a as opposed to the visually correct view

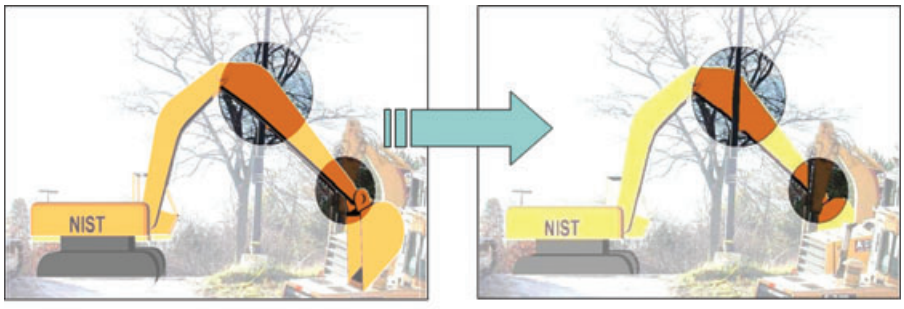


Fig. 1. Example of occlusion in an AR scene.

Table 1
Mechanisms for handling occlusion in different display systems

Display system	*Virtual occluding real*	*Real occluding virtual*
Screen based or back projection (CAVE)	Impossible	Inherent
Semi-transparent HMD	Inherent	Semi-visible Software solvable
Video see-through HMD	Inherent	Software solvable

shown in Figure 1b. This occurs because visual occlusion cannot be automatically handled and corrected unless appropriate methods are designed and integrated into the AR application that is generating the animation being viewed.

In fact, in all traditional AR applications, the real world is captured and displayed in the background although all virtual CAD objects displayed on the foreground cause the final display to be unable to depict the correct occlusion effect because the two groups of objects are completely separated (Breen et al., 1995; Wloka and Anderson, 1995). As a result, automated real-time occlusion handling becomes a critical step in animating dynamic simulation models in AR. The properties of the AR display system influence the approach to correctly handling occlusion (Taylor et al., 2007; Fuhrmann et al., 1999). As shown in Table 1, in screen-based or projection-based environments such as the Cave Automatic Virtual Environment (CAVE) (which represent a purely virtual immersive environment), handling occlusion of virtual objects by real objects is relatively simple and straightforward because real objects are always between the display surface and the observer's eyes and therefore always occlude virtual objects. However, virtual objects cannot occlude real objects because virtual graphics are always shown on the background screens although all real objects are located between the observer and the virtual background. When using a head-mounted display (HMD) to observe the augmented scene (e.g., in this research), the display surface is always between the observer's eyes and real objects, and the virtual objects occlude real objects by default. As a result, additional steps are required to handle cases in which real objects must occlude virtual CAD objects. In the presented research, depth and frame buffer manipulation techniques were used to develop a new automated algorithm for handling occlusion correctly. The presented approach is unique because it can be easily integrated into any mobile AR platform (such as UM-AR-GPS-ROVER introduced in Behzadan et al., 2008) that allows the observer of the AR animation to navigate freely in a scene and observe the animated graphics from different positions. The presented method is capable of automatically resolving occlusion effects in real time to produce visually convincing representations of the operations being animated.

2 PRIOR WORK IN OCCLUSION HANDLING

Several researchers have demonstrated algorithms and methods to correctly handle occlusion effects in AR. For example, Breen et al. (1995) presented techniques for interactively performing occlusion and collision detection between static real objects and dynamic virtual objects in AR. They used computer vision algorithms to acquire data that model aspects of the real world in the form of geometric models and depth maps. Wloka and Anderson (1995) presented a video see-through AR system capable of resolving occlusion between real and computer-generated objects. The heart of their system was an algorithm that assigns depth values to each pixel in a pair of stereo-video images in near real time. However, the use of stereo cameras caused their method to have difficulties in computing the depth for featureless (evenly lit, nontextured, and horizontal) rectangular image areas.

Lepetit and Berger (2000) introduced a semi-automatic approach to resolve occlusion in AR systems. Using their approach, once the occluding objects have been segmented by hand in selected views called "key frames," the occluding boundary is computed automatically in the intermediate views. To do that, the 3D reconstruction of the occluding boundary is achieved from the outlined silhouettes. Fischer et al. (2003) presented an algorithm based on a graphical model of static backgrounds in the natural surroundings, which has to be acquired beforehand. This algorithm is unable to deliver actual depth information for the scene. As a result, the main assumption was that whenever a real occluder is detected in front of the background, it is in front of all virtual objects. Hence, their method is primarily suitable for interaction with the AR scenes using hands or pointing devices, which can mostly be assumed to be closer to the user than virtual objects.

More recently, Feng et al. (2006) designed an optical-based algorithm to realize multilayer occlusion in indoor areas with objects only a few meters away from the viewer. The result of their work, however, caused unstable scenes as the process of object extraction was very sensitive to the change of ambient light illumination of the environment. Fortin and Hebert (2006) investigated model-based and depth-based approaches. Although the former is only suited for a static viewpoint and relies on a tracked bounding volume model within which the object's silhouette is carved, the latter makes

it possible to change the viewpoint by exploiting a handheld stereo camera. There are some limitations to the depth-based approach mainly due to the performances of local stereo algorithms. If the texture of the object is too uniform, the dense stereo correspondence may not be possible or at least is unreliable. In doubtful cases, such correspondences will be trimmed by filtering, leaving holes in the disparity map, which can result in missing 3D information for some areas of the real objects.

3 MAIN CONTRIBUTIONS

Most work conducted in occlusion handling thus far does not take into account the dynamics of the real world in which an AR animation takes place. Although some of them (Wloka and Anderson, 1995; Feng et al., 2006) use techniques that are most suitable for indoor controlled environments, several others (Breen et al., 1995; Lepetit and Berger, 2000; Fischer et al., 2003; Fortin and Hebert, 2006) use simplifying assumptions about the shape and position of real objects and the viewpoint from which the scene is observed that does not support the dynamic nature of objects in real world and the fact that their actual shape, position, and orientation can vary over time. For example, although the application of stereo cameras is an attractive option for real-world depth acquisition, the result is largely dependent on the nature of real objects, their physical characteristics and appearance, and distances they are located from the observer of the scene.

Producing correct occlusion effects in real time in an outdoor unprepared environment such as a construction site was the overall primary goal in developing an automated occlusion handling method in this research. The attained results had to be convincing enough to the observer of the scene. At the same time, no additional constraints over the user's maneuvering ability as well as position and orientation of both groups of real and virtual groups of objects could be created in the AR application. In addition, the required hardware components must be such that they do not limit the mobility of the AR platform due to factors such as heavy weight, dependence on ground power source, special care and maintenance, and user ergonomics.

Considering all these criteria, this research investigated approaches to develop an automated occlusion handling algorithm that would use real-world depth map input obtained through a remote-sensing device such as a laser detection and ranging (LADAR) camera (Langer et al., 2000; Wijesoma et al., 2004; Park et al., 2007). This device is connected alongside the video camera mounted on the observer of an AR scene. As discussed ahead in this article, the main advantages of flash LADAR devices are their light weight, high data resolution, and ability to extract depth data in almost any environment including outdoor construction sites. The limitations are constant noise in the incoming data, and limited operational range (on average less than 10 m). However, the occlusion handling method developed in the presented research has been designed to be generic and scalable so that future hardware with higher levels of accuracy and wider operational range can be easily plugged and used in an AR platform without any modifications or changes necessary to the core algorithms. Thus, the main contribution of the research presented in this article is a general-purpose occlusion handling algorithm integrated into a core AR platform capable of detecting and correctly resolving occlusion effects in real time. This method imposes no constraints over the position of real and virtual objects in an augmented scene, and the user can navigate freely within the augmented space in real time.

4 LEVEL OF DETAIL IN DEPTH ACQUISITION

Obtaining depth values for real and virtual objects is the most important step in correctly handling occlusion effects in AR. An intuitive approach to calculate the depth of a virtual CAD object in the scene is to extract its position in the Z direction (positive direction pointing straight at the user) because the graphical engine of the AR application (e.g., implemented in OpenGL; SGI, Sunnyvale, California) automatically keeps track of the depth of CAD objects relative to the viewpoint (user's eyes). The depth values for real objects, however, have to be acquired and recorded using more complex methods (described in Section 5) as they are not interpretably modeled in the computer. Because the incoming video stream of the real world is captured by and displayed in a monoscopic visual system in this research, the computer's knowledge of a real scene is limited to the plain video captured by the video camera without any depth information.

The level of detail according to which the depth of real and virtual objects is determined depends on the nature and characteristics of the objects (e.g., shape, surface material, and motion) in the scene. For a scene consisting of only a few simple static geometrical primitives, measuring the depth values at the object level seems to provide satisfactory results. Based on this approach, the depth of an object is represented by the distance between the user's eyes and a single point (or a few points) on that object. Selecting the most appropriate point on each object so that the depth is calculated as accurately as possible is a major challenge in this

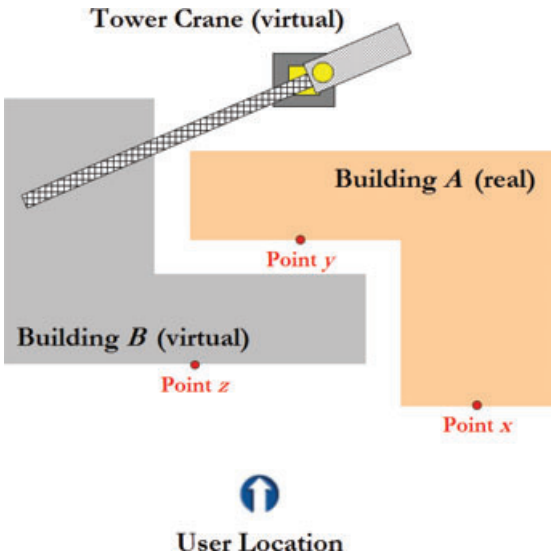
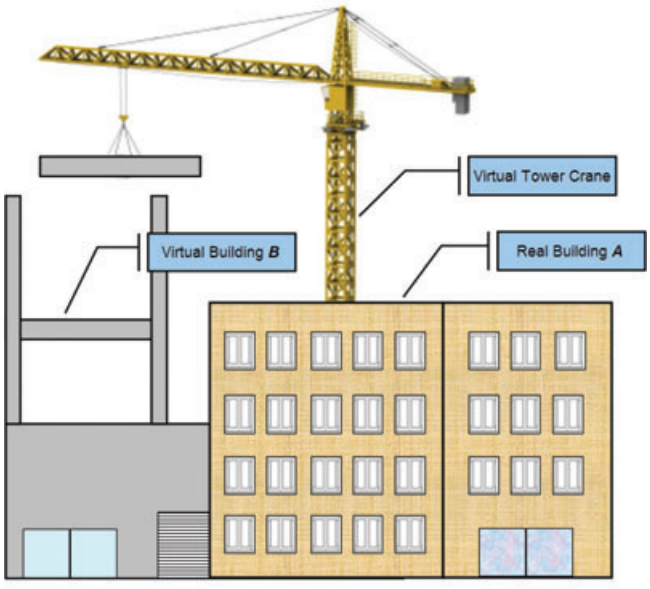


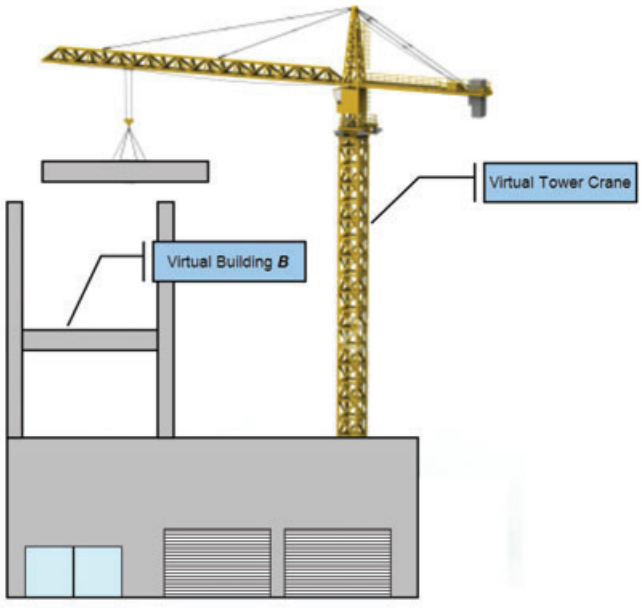
Fig. 2. Example of an object level of detailed depth calculation.

approach. Under some conditions, such as that shown in Figure 2, however, finding such a point becomes impossible and as a result, more than one point has to be picked for depth calculations. In this figure, a steel erection operation using a virtual tower crane to construct a virtual service building (i.e., building B) in front of a real residential building (i.e., building A) is shown. Following the object level of detailed approach described above and based on the location from which the user is observing the scene, there are two possible positions to pick depth representative points on building A (i.e., points x and y). In addition, point z can be selected to determine the depth of the virtual building.

Due to the way the scene is set up, choosing only one point on the real building (either point x or point y) is not enough for correct depth calculations. Having selected point x as the depth representative point for the real building A, because this point is closer to the user compared to point z (depth representative point for the virtual building), building A has to completely block building B in the user's view of the augmented scene. If point y is selected to represent the depth of the real building, the virtual building will completely block the user's view of the real building as point z is closer to the user's eyes compared to point y. As shown in Figures 3a and b, none of these two cases are visually correct and convincing enough to the user of the augmented scene. For the specific case of Figure 2, simultaneous selection of point x and point y on the real building A will correctly resolve the occlusion effect. In fact, the geometry of building A has to be divided into two separate parts and the depth of each part has to be independently compared with the depth of building B to create



a) Real building incorrectly occluding virtual building



b) Virtual building incorrectly occluding real building

Fig. 3. Incorrect occlusion effects using object level of detail.

a correct occlusion effect as shown in Figure 4. In this figure, parts of the real building (represented by point y in Figure 2) are blocked by the virtual building although the remaining part (represented by point x in Figure 2) is not occluded in the augmented view of the scene. Table 2 shows how the selection of different depth representative points will affect the final augmented view in terms of convincing occlusion effects. This example

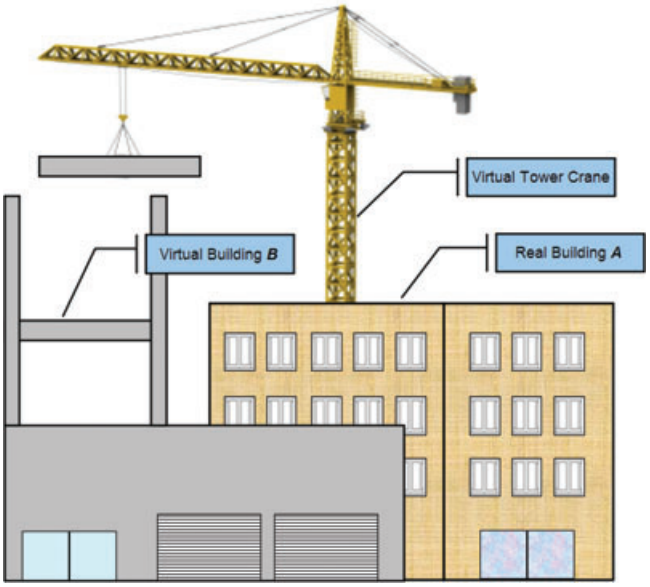


Fig. 4. Correct occlusion effect using multiple depth representative points.

Table 2
Effect of depth representative point selection on occlusion

Real building A		*Virtual building B*	*Occlusion effect*
x	*y*	*z*	
			N/A
•		•	Incorrect
	•	•	Incorrect
•	•	•	Correct

shows that objects operating in a dynamic environment such as a construction site can form a variety of scene setups. As a result, the process of point selection on all such objects becomes a significantly time-consuming task. Even if all such points are selected before the animation starts, there is no guarantee that the selected points can best represent the real scene and avoid further confusion in the depth calculation stage in cases such as that shown in Figure 3.

Another intuitive approach in resolving occlusion is to acquire and manipulate depth values at the polygon level of detail. Based on this approach, objects (real and virtual) are separated into smaller polygons. Rasterization algorithms are then applied to each pair of polygons to decide which has a lesser depth value and hence has to block the other. However, there are several cases in which it is impossible to make an accurate determination on which polygon is closer to the viewpoint. Figure 5 shows a scenario in which three polygons are to be placed in an animated scene. As shown in this figure, polygon A is closer to the viewpoint compared to polygon B, and polygon B is closer than polygon C. Based on these two observations, a default conclusion is that polygon A has to be closer to the viewpoint than both polygons B and C, which is clearly not the case in Figure 5.

To take into account all such uncertainties in the way the real world is set up, provide a sufficient level of detail capable of creating convincing resolved occlusion displays at an arbitrary viewing distance, and avoid unexpected depth miscalculation and paradox, a pixel level of detail was finally selected in this research. Working with pixels on an augmented screen ascertains a high degree of data resolution, which eliminates most of visual discrepancies that would have been created by a less accurate level of detail (i.e., object or polygon level of details). The depth acquisition and color manipulation algorithm designed in this research is shown in Figure 6. As shown in this figure, after the depth values for all real and virtual objects are obtained, the last step is to make a comparison at the pixel level to decide which object is closer to the user's eyes and hence has to be painted last. Following this approach, for a specific pixel on the screen, if the value of the real-world depth is less than that of the virtual world, a real object is occluding a virtual object. Further steps have to be then taken to update the color scheme of that pixel so that it is not painted in the color of the virtual object. These steps will be discussed in detail in Sections 5 and 6 of this article. Figures 7a and b show the graphical representation of the developed depth acquisition method for real and virtual objects, respectively. The specific pixel on the screen as shown in Figure 7 represents portions of a real tree and a virtual CAD model of an excavator. The depth of this pixel in the real world is captured using methods that will be described in Section 5. The depth value for the same pixel in the virtual world can be obtained using the transformation matrices and geometric properties of the CAD model represented by that pixel. Figure 8 illustrates the result of depth comparison performed at pixel level to determine whether or not the CAD object is occluded in the augmented view. The depth acquisition and color manipulation process depicted in Figures 6–8 has to be executed continuously by capturing the latest real and virtual depth values for all the pixels on the screen. Depth values of the real world can change if the viewpoint is moved (i.e., user changes position and/or head orientation) or there is a change in the contents of the real scene. Depth values of the virtual world can also change if CAD objects move in the scene. By constantly comparing these two sets of values, the depth effect can be included and represented in the augmented world to correctly resolve occlusion effects.

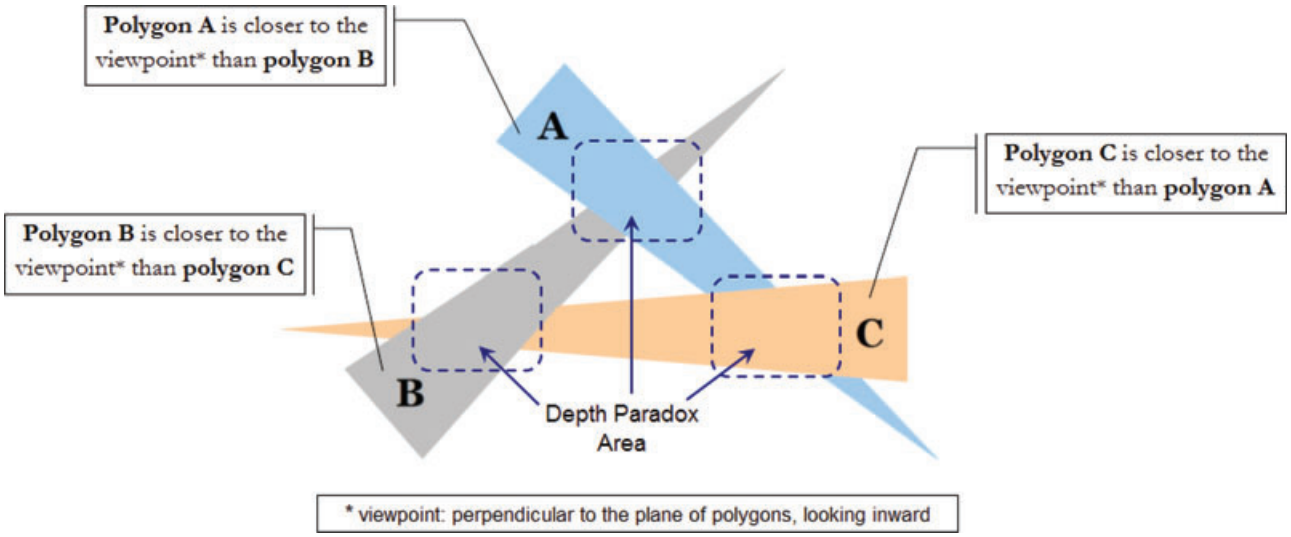


Fig. 5. Impossible occlusion handling case using polygon level of detail.

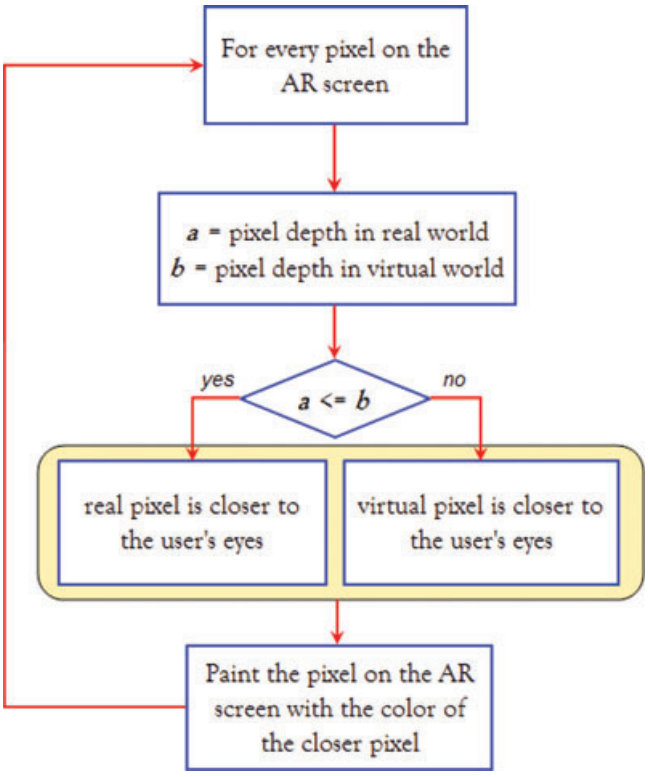
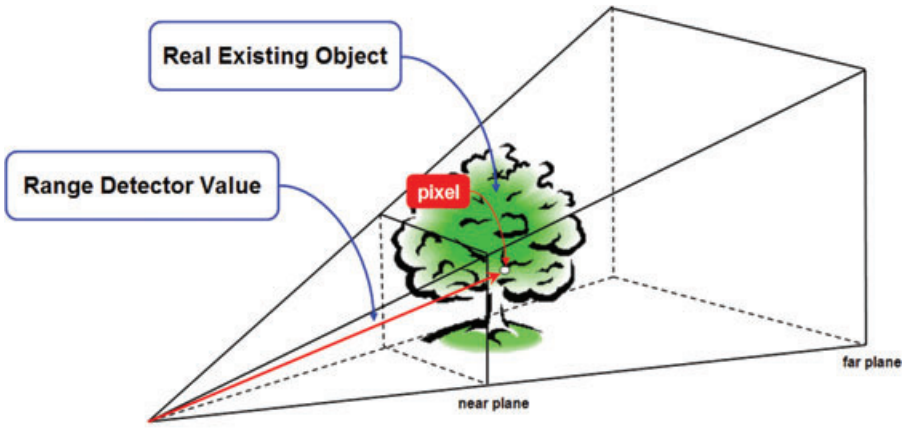


Fig. 6. Designed depth acquisition and color manipulation algorithm.

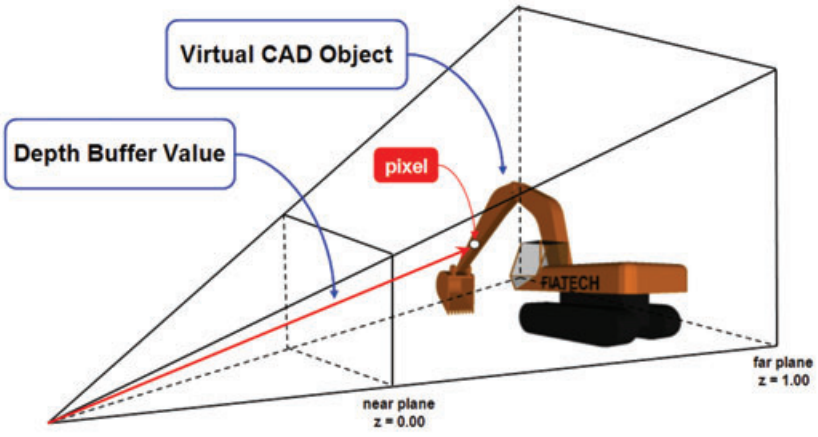
5 DEPTH SENSING AND CONVERSION TECHNIQUES

Producing correct occlusion effects in real time in an outdoor unprepared environment such as a construction site was a primary concern in developing the automated occlusion handling method in this research. The authors have successfully designed and implemented a mobile computing apparatus equipped with components necessary to perform a walk-through AR animation in real time (Behzadan et al., 2008). As shown in Figure 9, the apparatus takes advantage of real-time positioning data coming through a GPS receiver as well as 3D head orientation data supplied by a head orientation tracker device (inside the hard hat) to position the user inside the AR animation. In the mobile computing apparatus shown in Figure 9, the main computing task is performed by a laptop computer secured inside the backpack. Although the real scene is captured by a video camera in front of the user's eyes, the rendered graphical scene is displayed to the user through the HMD installed in front of the hard hat. At the same time, the user can interact with the system using a miniature keyboard and a touchpad. Although the resulting AR animation has to be convincing enough to the observer of the scene, no additional constraint over the user's maneuvering ability as well as position and orientation of both groups of real and virtual objects has to be imposed by the AR application. In addition, the required depth-sensing hardware components to perform the task of occlusion handling have to be selected in a way that they do not limit the mobility of the AR platform due to factors such as heavy weight, dependence on ground power source, special care and maintenance, and user ergonomics.

As noted earlier, depth acquisition at pixel level of detail has the advantage that the scene can be arbitrarily complex, although the processing time remains a constant time function of image resolution. Additionally, no geometric model of the real environment is needed



a) Relative location of real object in depth map

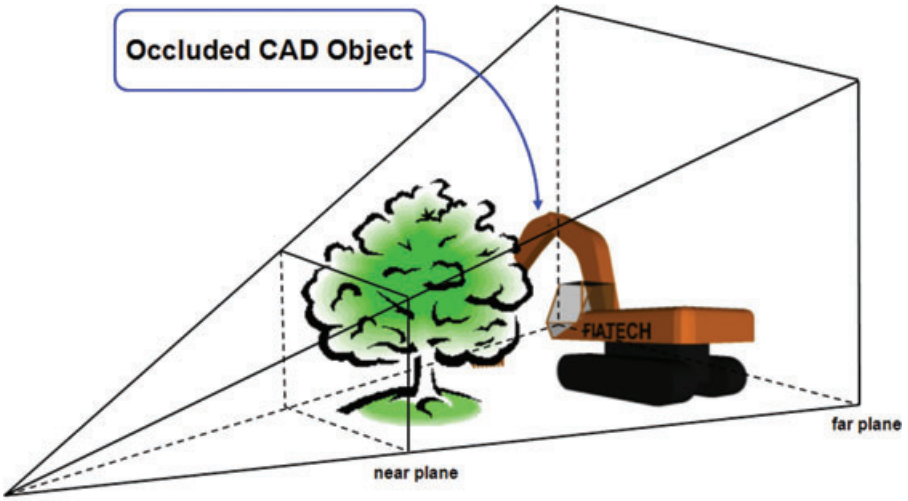


b) Relative location of virtual object in depth map

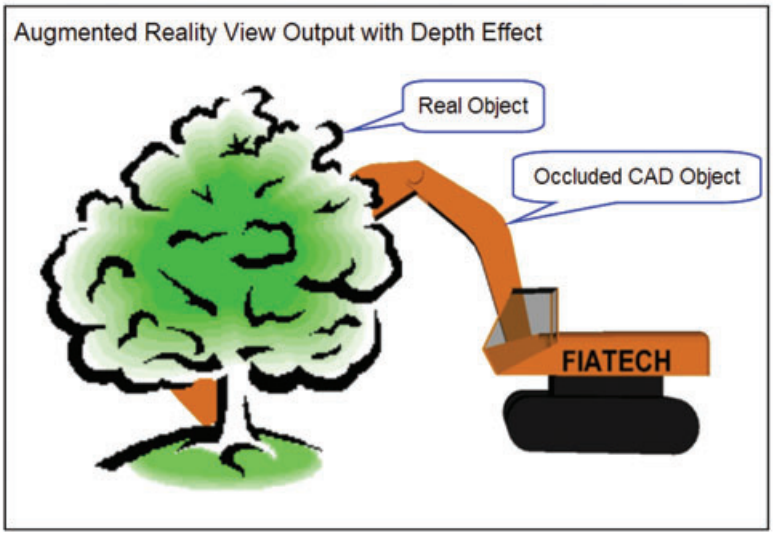
Fig. 7. Capturing the depth of real and virtual objects.

during the animation. However, the depth map is dependent on the user's position and head orientation, as well as the location of real objects in the scene. Once the user or the real objects change their position and/or orientation, the depth map becomes invalid (Breen et al., 1995). To take into account all such variations, the process of depth acquisition and comparison has to be done in real time. The hardware components required to perform depth acquisition have to be selected in a way that they can be easily integrated into any existing mobile AR platform. Hence, being lightweight and self-powered, having a convenient interface, and supporting an acceptable pixel resolution are among the important factors in selecting the hardware components. There is certainly a tradeoff between equipment mobility and data resolution. The heavier a camera is, for example, the more data sample points it can collect and the resulting image is more accurate. This directly translates into more processing time that is not desirable for the purpose of this research as all calculations have to be performed in real time. Lighter cameras, although providing lower resolution depth images, can operate at a faster processing speed and hence are better fits for the AR platform used in this research.

Several options were studied including high-resolution cameras such as 3DLS (7–8 kg) (Fraunhofer IAIS, Sankt Augustin, Germany), Konica Vivid 9i (15 kg) (Konica Minolta Business Solutions U.S.A. Inc., Ramsey, New Jersey), I-Site 4400 LR (14 kg) (Maptek USA, Lakewood, Colorado), FARO LS 420 (14.5 kg) (FARO, Coventry, United Kingdom), and Leica HDS 3000 (17 kg) (Leica Geosystems AG, St. Gallen,



a) Relative location of real and virtual object in depth map



b) Resulting occlusion in planar representation

Fig. 8. Final AR screen with correct occlusion effect.

Switzerland). Although all these cameras provide full range data (wide horizontal and vertical scanning angles), they cannot be mounted on mobile platforms mainly due to their weight and dependence on external power sources. In addition, according to most manufacturers, fixed tripods have to be used as mounting bases to achieve best performance. Another category of imaging cameras is lightweight cameras such as flash LADAR devices that are typically suitable for low-range applications. This category of cameras is more suitable for this research to perform real-world depth acquisition as it uses a promising technology providing robust and accurate access to depth data of the real objects, and has already proved to provide satisfactory results in geometric modeling of construction sites for automation and robotics applications (Teizer et al., 2005a, b, 2007). Table 3 shows some details of three such cameras that are most used in relevant research projects.

Fig. 9. Profile of a user equipped with mobile AR computing apparatus.

A flash LADAR system typically consists of a device constantly casting laser beams in the 3D real space and receiving the resulting reflected beams. Based on the travel time for each beam and knowing the speed of the laser beam, the distance between the flash LADAR system and each real object in the scene is calculated. Once installed in front of the user's eyes, these depth values reflect the distance between the viewpoint and the real objects. Flash LADAR data represents a scene as a matrix. Each element in this matrix contains the depth value of the corresponding pixel on the screen. As a result, the concept of *screen matrix* was introduced and used in this research to provide a means to store and retrieve depth values more efficiently inside the AR platform. By definition, a screen matrix is a 2D matrix with dimensions equal to the pixel resolution of the screen. Each element in this matrix can hold data [e.g., Red-Green-Blue (RGB), color and depth value] about the corresponding pixel on the screen.

Figure 10 shows a sample screen matrix for a 640-by-480 screen. As noted earlier, the elements in this matrix can store depth values as well as color codes for their corresponding pixels on the screen. If the resolution of the depth data obtained from a flash LADAR device is less than the resolution of the actual screen, adjacent elements of the screen matrix can be clustered together and a single depth value (obtained from the flash LADAR device) can be assigned to all the elements inside a cluster. For example, if the incoming flash LADAR depth data have a resolution of 160 × 120 although the actual screen resolution is 640 × 480, four adjacent pixels (in both horizontal and vertical directions) can be grouped as one cluster and the same depth value can be assigned to all of them. Depth values for the pixels on the virtual screen are also retrieved from the OpenGL z-buffer. The z-buffer is a memory buffer in the OpenGL graphics accelerator that holds the latest depth of each pixel along the Z-axis. As the virtual contents of the scene change over time, the values stored in the z-buffer are also updated to reflect the latest depth of the CAD objects relative to the viewpoint (i.e., user's eyes).

The fundamental difference between depth values obtained from a flash LADAR camera for real objects and those obtained from z-buffer for virtual objects is that although real-world depth values are retrieved and reported as real distances (in terms of meters or feet) to the user, depth values for the virtual CAD objects fall

Table 3
Manufacturer's properties of different flash LADAR devices

	Dimension (cm)			*Pixel resolution*		*Field of view (°)*			
Model	*x*	*y*	*z*	*H*	*V*	*H*	*V*	*Frame rate*	*Range (m)*
CSEM SR3000[1]	5.00	6.70	4.23	176	144	39.6	47.5	50	7.5
CSEM SwissRanger2[2]	14.60	3.10	3.30	160	124	42.0	46.0	30	7.5
PMD 19K[3]	20.80	17.40	4.40	160	120	40.0	30.0	15	5.0 ~ 30.0

[1]Mesa Imaging AG, Zurich, Switzerland.
[2]CSEM SA, Zurich, Switzerland.
[3]PMD Technologies GmbH, Siegen, Germany.

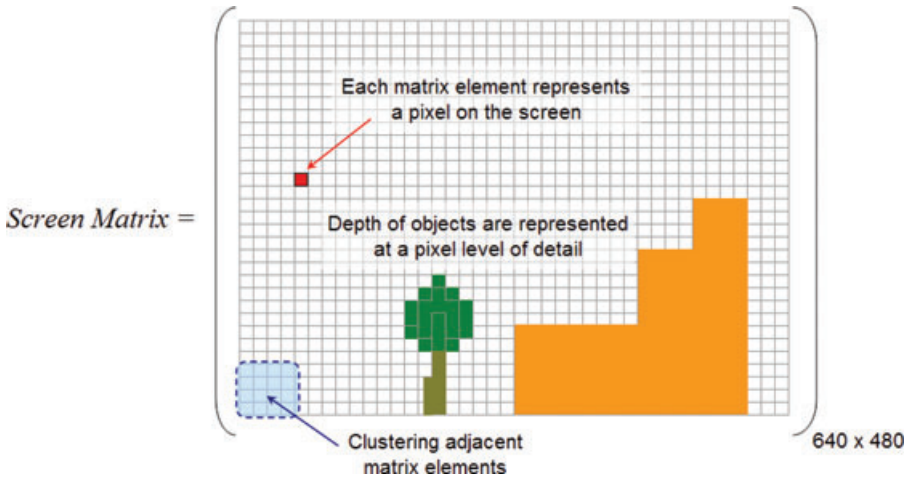


Fig. 10. Sample screen matrix for a 640-by-480 screen.

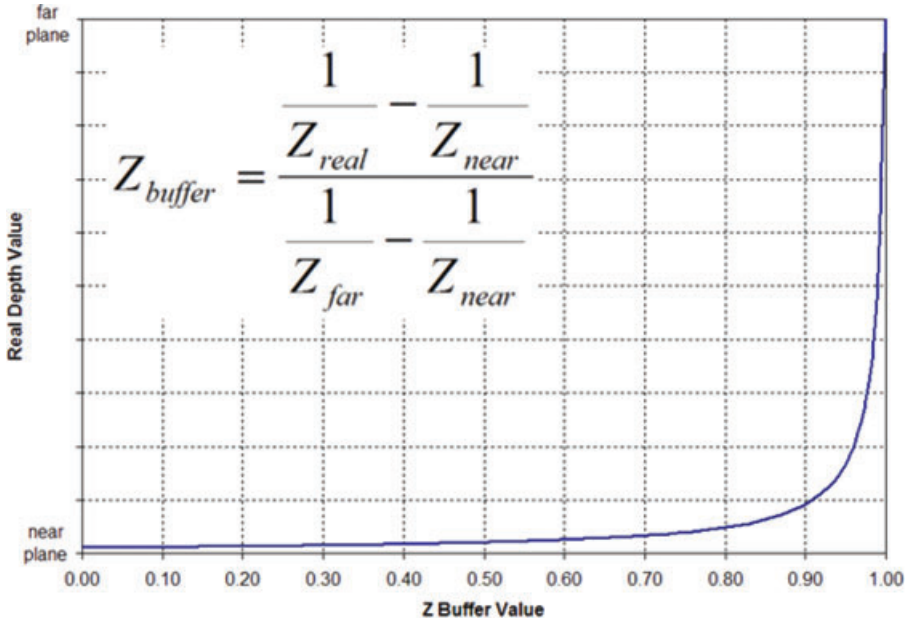


Fig. 11. Relation between z-buffer and metric virtual depth values.

between a [0, 1] interval. A pixel located on the near plane of the perspective viewing frustum will be given a depth value equal to zero, and a pixel located on the far plane of the perspective viewing frustum will be given a depth value equal to one. All intermediate pixels will have depth values between zero and one. However, the relationship between the depth values obtained from the z-buffer and corresponding metric depth values is not linear. A pixel with a virtual depth value of 0.5 is not located halfway between the near and far planes. In fact, this relationship, as shown in Figure 11, follows a hyperbolic equation (Fortin and Hebert, 2006). In this figure, Z_{near} and Z_{far} correspond to the metric distance between the user's eyes and the near and far planes of the perspective viewing frustum, respectively. Z_{buffer} is the depth value of a specific pixel on the screen obtained from the z-buffer and Z_{real} is the metric equivalent of this depth value for the same pixel. As shown in Figure 11, z-buffer has higher resolution for pixels closer to the user's eyes. For the specific case shown in Figure 11, more than 90% of all possible z-buffer values represent the depth of objects located within 10% of the distance between the near and far planes. This is mainly because the human eyes are more sensitive to closer

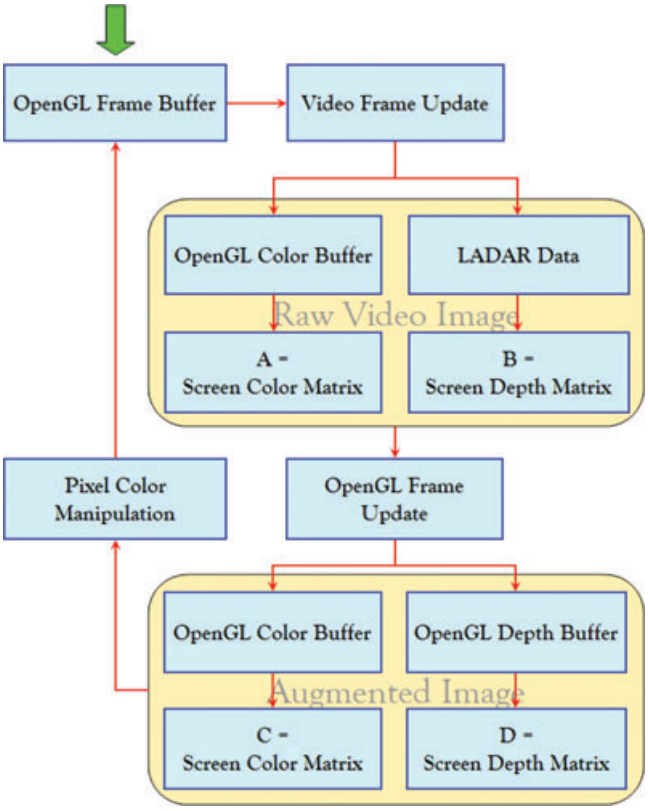


Fig. 12. Constructing four distinct color and depth matrices.

objects and can identify any short-range visual discrepancies more rapidly, whereas discrepancies in objects that are farther away cannot be recognized as clearly as in closer objects. Depth values for the virtual objects are also stored in a separate screen matrix for later comparison with corresponding values of the real world.

6 FRAME BUFFER MANIPULATION

After all depth values are obtained and stored appropriately in separate screen matrices, they have to be compared so that for each pixel a determination is made on which group of objects (real or virtual) is closer and hence has to be displayed. Once this is computed, the color of the pixel will be changed to the color of the closer object to the viewpoint to create the impression that the pixel really represents the correct object. This requires an intermediate step, that is, obtaining the color values (in terms of RGB) for each individual pixel on the screen. This can be done by directly reading the OpenGL color buffer that stores pixel color values in real time. Two distinct readings are done to obtain pixel colors of the scene with and without virtual objects. Once depth and color readings are complete, four different matrices are provided to the AR application:

- A = Screen matrix of real-world color values (captured texture from the video camera as stored in OpenGL color buffer before superimposition of CAD models)
- B = Screen matrix of real-world depth values (representing the depth of the raw video input coming through the video camera)
- C = Screen matrix of CAD models' color values (OpenGL color buffer after superimposition of CAD models)
- D = Screen matrix of CAD models' depth values (OpenGL depth buffer after superimposition of CAD models)

Figure 12 shows how these four matrices are constructed using the contents of depth and color buffers as well as the incoming data through depth-sensing hardware (e.g., flash LADAR device). As shown in this figure, starting from the top left-hand corner, after the OpenGL frame buffer is refreshed, the content of the video frame is updated using the raw video image coming through the video camera. At this time, matrix A is constructed using pixel color values of the raw video image and matrix B is constructed using the captured depth data of the real world. Once the OpenGL frame is updated, the virtual contents of the scene are displayed on top of the real background. At this point, matrix C is constructed using the contents of the OpenGL color buffer and matrix D is constructed using the OpenGL z-buffer values. All these operations occur at each animation frame to handle occlusion continuously as the animation is running.

Figure 13 shows the frame buffer manipulation algorithm designed in the presented research, which uses these four matrices to construct a final screen matrix (i.e., matrix E in this figure). Screen matrix E contains correct pixel colors after resolving all incorrect occlusion cases. In this figure, for every pixel on the screen, the corresponding color and depth value is read from the four matrices described above (shown as *a*, *b*, *c*, and *d* in Figure 13). The real and virtual depth values are then compared. If the depth of the pixel in the real world is less than its depth in the virtual world, its color is changed to the color read from the matrix representing real-world colors (i.e., matrix A). This represents the case in which a real object is occluding a virtual object. The other situation occurs when the depth of the pixel in the virtual world is less than its depth in the real world. This represents the case in which a virtual object is occluding a real object and hence the pixel color is changed to the color read from the matrix representing virtual world colors (i.e., matrix C). The correct

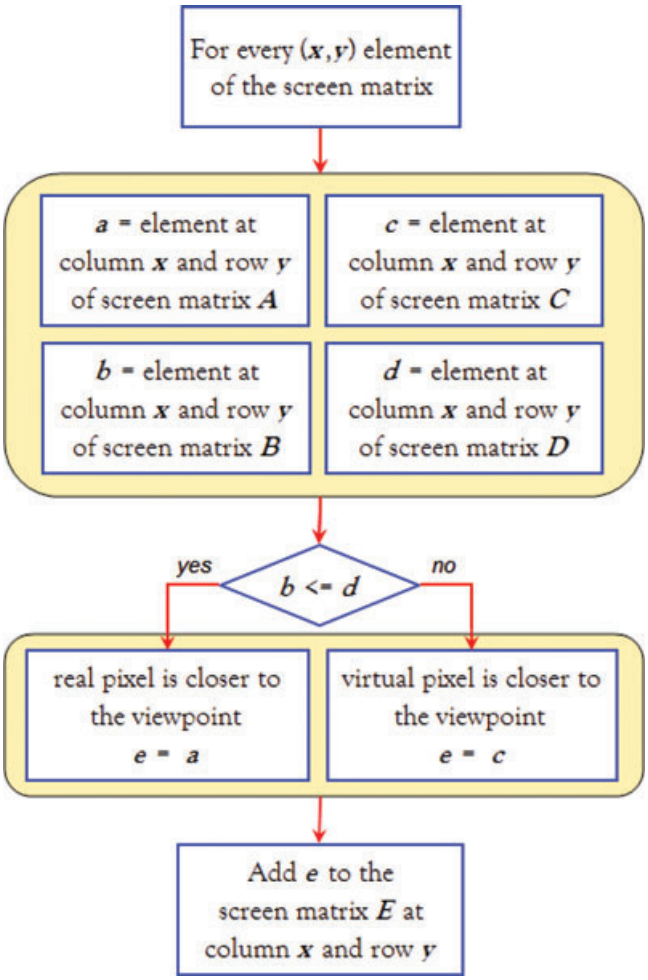


Fig. 13. Designed frame buffer manipulation algorithm.

pixel color (shown as *e* in Figure 13) is then stored in the screen matrix E, which will be later used when the OpenGL frame buffer is updated to show the correct occlusion effect.

7 VALIDATION

Several proof-of-concept experiments were conducted to validate the functionality of the frame buffer manipulation algorithm developed in this research. The objective of the experiments was to validate that the designed occlusion handling algorithm is capable of detecting and resolving visual occlusion cases in a real-time AR animation and produce visually convincing output representing the modeling operation. Animation script files of small-scale construction operations were created using the ARVISCOPE language (Behzadan, 2008; Behzadan and Kamat, 2009a). Miniature models of actual construction equipment and materials were used to create the real background of the augmented animation. Small-scale CAD models were then superimposed on the real background to create the final augmented view of the operations. In each experiment, the distance between the virtual CAD objects and the user was set to be greater than that of the real construction objects to verify that the developed algorithm was capable of detecting and handling incorrect occlusion cases. As discussed earlier, the opposite case in which the distance between a virtual object and the viewpoint is less than that of a real object is trivial and is automatically taken into account by almost any AR-based visualization application.

Following the range data of typical LADAR devices shown in Table 3, and in order not to be confined in a limited operational range when conducting the experiments, depth data of real objects used in the validation were obtained from physical measurements taken around the layout of the real objects. This, in fact, provided a wider experiment range, which enabled the viewpoint to be set up at a farther location inside the augmented environment. As described later in the next section, using physical measurements to obtain real-world depth data does not contradict with the fact that the designed occlusion handling algorithm is able to compare depth values in real time to effectively detect and correct visual occlusion cases at the pixel level of detail. In contrast, it supports the idea that the developed occlusion handling algorithm is generic enough that it can handle depth data from a variety of sources (e.g., manual measurements, LADAR devices, and 3D laser scanners) to perform the basic task of occlusion handling. This is a significant feature that makes the presented technique independent of limitations in resolution and range introduced by available market products.

Figures 14–16 show results of three validation experiments conducted in this research. In each experiment, CAD models of construction equipment and machinery were superimposed on top of miniature construction environments consisting of real scaled construction models. Virtual models were placed in the augmented scene in a way that they were completely or partially occluded by real objects. Figure 14 shows a virtual excavator occluded by a real structure. In Figure 15, a virtual dozer is partially occluded by a real tower crane. The virtual forklift in Figure 16 is occluded by a real container. As observed in these figures, the occlusion was accurately detected and resolved in all cases using the designed occlusion handling algorithm.

Validating the results of this research in larger scale industrial applications requires more powerful and accurate depth-sensing devices to be invented and

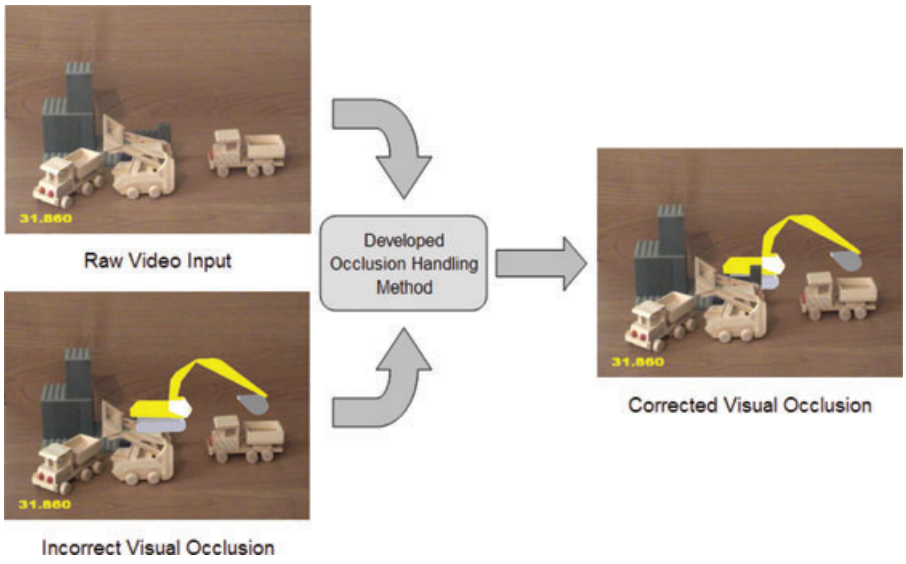


Fig. 14. Correcting occlusion between a virtual excavator and a real structure.

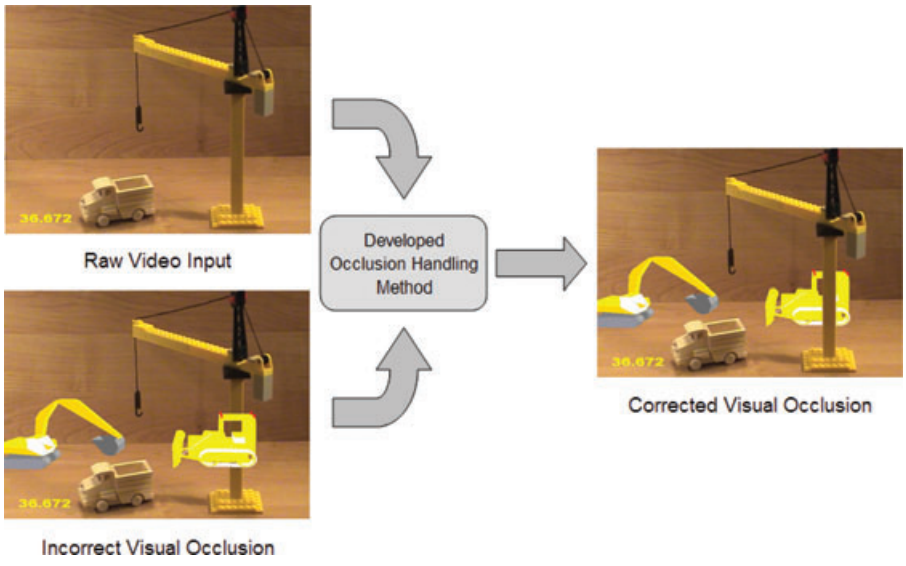


Fig. 15. Correcting occlusion between a virtual dozer and a real tower crane.

introduced by the market. Such devices must be able to operate under harsh conditions that a typical construction project is subject to. Factors such as the intensity of ambient light and laser beam reflection on metallic surfaces are major drawbacks for the application of existing LADAR devices in the field of construction where the majority of work is performed in outdoor environments. Meanwhile, the authors are continuously working to prepare the required infrastructure for future implementations. For example, the authors are currently working on an ongoing research project, which is extensively focused on the application of AR and occlusion handling in excavation projects and prevention of physical damage to underground utilities (Behzadan and Kamat, 2009b). In another research project, the authors are exploring methods of equipping heavy construction machinery with motion sensors and laser scanners to track their motion in real time and produce augmented views of the surrounding construction environment in context to the equipment operators and site engineers for control and inspection purposes.

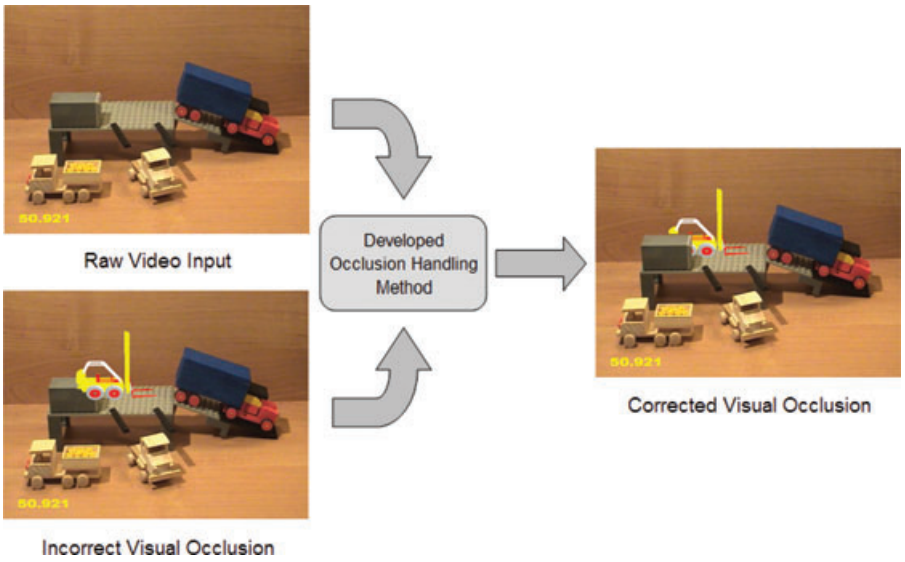


Fig. 16. Correcting occlusion between a virtual forklift and a real container.

8 SUMMARY AND CONCLUSIONS

In this article, a depth-based occlusion handling method was introduced that enables AR visualization tools to sense depth data of virtual and real objects and detect and correct occlusion cases between virtual and real objects in real time. Although AR has been recently used to address visualization needs for indoor small-scale applications under controlled environments, lack of adequate and robust solutions to problems such as incorrect occlusion that arise from the dynamic nature of objects in an outdoor unprepared environment have been a major challenge in developing and implementing functional AR-based animations for outdoor larger scale tasks (e.g., construction operations). In fact, unless appropriate object depth detection methods with sufficient level of detail are deployed, AR animations are unable to correct visual occlusion cases that happen when a real object is closer to the observer but is visually blocked by a virtual CAD object that is intended to be farther away. The level of detail in occlusion handling is heavily a function of the degree of complexity and uncertainties involved in the visualized augmented scene. For example, a dynamic environment such as a construction jobsite consists of a large number of real objects (personnel, equipment, and material) continuously changing shape, position, and orientation. Once a simulated operation is superimposed over the existing real display, the scene dynamics change as real objects not only interact with each other but also have to maintain a close logical and spatial relation with virtual objects.

The occlusion handling method and AR visualization tool described in this article are capable of detecting, resolving, and displaying correct occlusion effects, using depth and frame buffer manipulation techniques at the pixel level of detail in dynamic augmented environments. The developed approach is unique because it can be easily integrated into any mobile AR platform, which allows the observer of an AR animation to navigate freely and observe the ongoing operations from different perspectives. Several proof-of-concept experiments were conducted to validate the functionality of the developed occlusion handling method. In particular, scaled models of actual construction equipment and materials were used to create the real background of the augmented animation, and small-scale CAD models were superimposed on the real background to create the final augmented view of the operations although the occlusion cases were correctly resolved in real time. The authors fully acknowledge the fact that further improvement in this field is heavily dependent on available market products that provide more data resolution and higher operational range. This has been reflected in their work by designing an occlusion handling method that is generic in both concept and design. Although the implementation stage presented in this article mainly illustrated the results of a series of proof-of-concept experiments, the developed depth sensing and frame buffer manipulation techniques do not impose any limitations on the actual depth-sensing device that may eventually become available in the market.

ACKNOWLEDGMENTS

The presented work has been supported by the National Science Foundation (NSF) through grant CMS-0448762. The authors gratefully acknowledge NSF's support. The authors thank Professor Klaus-Peter Beier at the University of Michigan for his invaluable insight and advice, as well as Ph.D. student Ms. Sara Jabbarizadeh for her assistance in conducting the described experiments. Any opinions, findings, conclusions, and recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF or the individuals mentioned above.

REFERENCES

Barnes, M. R. (1997), An Introduction to QUEST, in *Proceedings of Winter Simulation Conference (WSC)*, IEEE, Atlanta, GA, 619–23.

Behzadan, A. H. (2008), ARVISCOPE: Georeferenced visualization of dynamic construction processes in three-dimensional outdoor augmented reality, PhD Dissertation, University of Michigan, Ann Arbor, MI.

Behzadan, A. H. & Kamat, V. R. (2007), Georeferenced registration of construction graphics in mobile outdoor augmented reality, *Journal of Computing in Civil Engineering*, **21**(4), 247–58.

Behzadan, A. H. & Kamat, V. R. (2009a), Automated generation of operations level construction animations in outdoor augmented reality, *Journal of Computing in Civil Engineering*, Special Issue on Graphical 3D Visualization in AEC, American Society of Civil Engineers (ASCE), Reston, VA, in press.

Behzadan, A. H. & Kamat, V. R. (2009b), Interactive augmented reality visualization for improved damage prevention and maintenance of underground infrastructure, in *Proceedings of the 2009 Construction Research Congress*, Seattle, WA.

Behzadan, A. H., Timm, B. W. & Kamat, V. R. (2008), General purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering, *Journal of Advanced Engineering Informatics*, **22**(1), 90–105.

Bishop, J. L. & Balci, O. (1990), General purpose visual simulation system: a functional description, in *Proceedings of the Winter Simulation Conference (WSC)*, IEEE, New Orleans, LA, 504–12.

Breen, D. E., Rose, E. & Whitaker, R. T. (1995), *Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality*, Technical Report ECRC-95-02, European Computer-Industry Research Center, Munich, Germany.

Brooks, Jr., F. P. (1999), What's real about virtual reality? *Journal of Computer Graphics and Applications*, **16**(6), 16–27.

Dubois, E. & Nigay, L. (2000), Augmented reality: Which augmentation for which reality?, in *Proceedings of DARE 2000 on Designing Augmented Reality Environments*, ACM, Elsinore, Denmark, 165–66.

Feng, Y., Du, W., Guan, X., Gao, F. & Chen, Y. (2006), Realization of multilayer occlusion between real and virtual scenes in augmented reality, in *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*, Nanjing, China, 1–5.

Fischer, J., Regenbrecht, H. & Baratoff, G. (2003), Detecting dynamic occlusion in front of static backgrounds for AR scenes, in *Proceedings of the Workshop on Virtual Environments*, Zurich, Switzerland, 153–61.

Fortin, P. A. & Hebert, P. (2006), Handling occlusions in real-time augmented reality: dealing with movable real and virtual objects, in *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, Quebec City, QB, Canada, 54.

Fuhrmann, A., Hesina, G., Faure, F. & Gervautz, M. (1999), Occlusion in collaborative augmented environments, *Journal of Computers and Graphics*, **23**(6), 809–19.

Gleue, T. & Dähne, P. (2001), Design and implementation of a mobile device for outdoor augmented reality in the archeoguide project, in *Proceedings of the 2001 Conference on Virtual Reality, Archeology, and Cultural Heritage*, ACM Press, Glyfada, Greece, 161–68.

Hegror, G., Palamidese, P. & Thalmann, D. (1989), Motion control in animation, simulation, and visualization, *Computer Graphics Forum*, **8**(4), 347–52.

Kamat, V. R. (2003), VITASCOPE: extensible and scalable 3d visualization of simulated construction operations, PhD Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Langer, D., Mettenleiter, M., Hartl, F. & Frohlich, C. (2000), Imaging ladar for 3-D surveying and CAD modeling of real-world environments, *Robotics Research*, **19**(11), 1075–88.

Lepetit, V. & Berger, M. O. (2000), A semi-automatic method for resolving occlusion in augmented reality, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, 2225–30.

Livingston, M., Rosenblum, L., Julier, S., Brown, D. & Baillot, Y. (2002), An augmented reality system for military operations in urban terrain, in *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC '02)*, National Training and Simulation Association (NTSA), Orlando, FL, 1–8.

Noma, H., Miyasato, T. & Kishino, F. (1996), A palmtop display for dexterous manipulation with haptic sensation, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground*, ACM, Vancouver, BC, Canada, 126–33.

Op den Bosch, A. (1994), Design/construction process simulation in real-time object-oriented environments, PhD Dissertation, Georgia Institute of Technology, Atlanta, GA.

Park, H. S., Lee, H. M., Adeli, H. & Lee, I. (2007), A new approach for health monitoring of structures: terrestrial laser scanning, *Computer-Aided Civil and Infrastructure Engineering*, **22**(1), 19–30.

Roberts, G. W., Evans, A., Dodson, A., Denby, B., Cooper, S. & Hollands, R. (2002), *The Use of Augmented Reality, GPS, and INS for Subsurface Data Visualization*, FIG XXII International Congress, Washington DC.

Rohrer, M. W. (2000), Seeing is believing: the importance of visualization in manufacturing simulation, in *Proceedings of the Winter Simulation Conference (WSC)*, IEEE, Orlando, FL, 1211–16.

Rohrer, M. W. & McGregor, I. W. (2002), Simulating reality using AUTOMOD, in *Proceedings of the Winter Simulation Conference (WSC)*, IEEE, San Diego, CA, 173–81.

Shreiner, D., Woo, M., Neider, J. & Davis, T. (2004), *OpenGL Programming Guide*, Addison Wesley, Reading, MA.

Taylor, N. R., Panchev, C., Hartley, M., Kasderidis, S. & Taylor, J. G. (2007), Occlusion, attention and object representations, *Integrated Computer-Aided Engineering*, **14**(4), 283–306.

Teizer, J., Caldas, C. H. & Haas, C. (2007), Real-time three-dimensional occupancy grid modeling for the detection and tracking of construction resources, *Journal of Construction Engineering and Management*, **133**(11), 880–88.

Teizer, J., Kim, C., Bosche, F., Caldas, C. H. & Haas, C. T. (2005a), Real-time 3D modeling for accelerated and safer construction using emerging technology, in *Proceedings of the 1st International Conference on Construction Engineering and Management*, Seoul, Korea, 539–43.

Teizer, J., Liapi, K., Caldas, C. & Haas, C. (2005b), Experiments in real-time spatial data acquisition for obstacle detection, in *Proceedings of the Construction Research Congress (CRC)*, San Diego, CA, 107–16.

Thomas, B., Close, B., Donoghue, J., Squires, J., Bondi, P., Morris, M. & Piekarski, W. (2000), ARQuake: An outdoor/indoor first person augmented reality application, in *Proceedings of the 4th International Symposium on Wearable Computers (ISWC2000)*, IEEE, Atlanta, GA, 139–46.

Webster, A., Feiner, S., MacIntyre, B., Massie, W. & Krueger, T. (1996), Augmented reality in architectural construction, inspection and renovation, in *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 913–19.

Wijesoma, W. S., Kodagoda, K. R. S. & Balasuriya, A. P. (2004), Load-boundary detection and tracking using ladar sensing, *Transactions on Robotics and Automation*, **20**(3), 456–64.

Wloka, M. M. & Anderson, B. G. (1995), Resolving occlusion in augmented reality, in *Proceedings of the Symposium on Interactive 3D Graphics*, Monterey, CA, 5–12.