Improving Obstacle Awareness to Enhance Interaction in Virtual Reality

Ivan Valentini  Giorgio Ballestin  Chiara Bassano  Fabio Solari  Manuela Chessa *

Dept. of Informatics, Bioengineering, Robotics, and Systems Engineering
University of Genoa, Italy

ABSTRACT
Currently, immersive virtual reality is experienced through head-mounted displays while the user is physically present into a real, cluttered environment. This causes the problem of avoiding dangerous collisions with obstacles in the real environment that are invisible to the user, and also hampers the interaction with real objects. Following the augmented virtuality paradigm, these obstacles should be embedded into the virtual environment. Thus, there is the need of knowing the 3D structure of the real environment to align it with the virtual one. In this paper, we present a method to create a virtual scenario composed of virtual objects having the same spatial occupancy of the corresponding real ones. The real scene is scanned to detect the position and bounding box of objects and obstacles, then virtual elements having similar spatial properties are added to the virtual scene. Two different real environment structure detection and clustering techniques are described and compared, both quantitatively and by considering users’ sense of presence with respect to the standard Chaperone technique. Our results show that the method is a good solution to maintain the real environment awareness while keeping an high level of immersivity and sense of presence.

Index Terms: Human-centered computing—Interaction paradigms—Virtual Reality; Human-centered computing—Interaction paradigms—Mixed / augmented reality;

1 INTRODUCTION
One of the main aims of immersive Virtual Reality (VR) is to create virtual environments able to elicit a strong sense of presence [14,16], defined as the sensation of being physically inside the simulated world, and with which the user can interact in a natural way.

When wearing head-mounted displays (HMDs), it is desirable not to be distracted by external factors, to retain a high degree of immersion. Nevertheless, having a knowledge of the surrounding environment can be important, from two points of view. First of all, one of the main drawbacks of wearing an HMD is that collisions with real-world objects invisible to the user can occur, thus creating serious physical problems to the user him/herself. On the other hand, it is possible to exploit the presence of physical real objects surrounding the HMD user, in order to augment the virtual experience itself. Real-space interaction is indeed one of the main goals of mixed reality (MR) and augmented virtuality (AV) setups [1, 10, 20]. The idea is to overcome the lack of tactile feedback of VR, by mixing real and virtual information and exploiting the physical objects existing in the scene.

The approaches present in the literature are different depending on the desired goal, such as avoiding obstacles or enabling physical interaction, but all of them have to preserve immersion and sense of presence in VR.

In this paper, we propose a method to address the problem by considering the two different aspects, and with the main goal of preserving the sense of presence.

We propose to: (i) reconstruct the 3D structure of the scene where the HMD is going to be used, to detect the objects in the field of action; (ii) create a virtual scene by embedding virtual objects, with size and position consistent with the real objects. In particular, in the specific implementation we are describing, we detect the size, orientation and 3D position of the objects. Moreover, virtual and real environments should be spatially aligned, in order to allow a coherent perception and interaction with obstacles.

The paper is organized as follow. In Section 1.1 we briefly describe the state-of-the-art, by discussing the features and the limits of the approaches, recently proposed in the literature. In Section 2, we describe the technical aspects of the proposed method. In Section 3 we evaluate the method, considering both to the accuracy of the virtual reconstruction and to the user experience, by addressing the sense of presence, in a preliminary user study. Finally, in Section 4 we discuss the open problems and the future developments of the described method.

1.1 State of the art
Since the occlusion of real-world objects is problematic for virtual reality systems users, several solutions attempt to address and solve the problem. Many of them add an out of context layer of information to the virtual environment influencing the immersivity and the sense of presence of the user [14–16].

The Chaperone system is a utility designed by Valve to be used with their VR platform, SteamVR 1, and the HTC Vive 2. Once set up, it keeps track of where a user is inside the tracked area. The main purpose of the system is to warn the users when they approach an obstacle that they cannot see because of the headset. Several possible settings help Chaperone being less intrusive in the virtual environment, but in any case its main drawback is that it breaks the sense of presence. The user in fact sees, in the virtual environment, the boundaries of the tracked area or the real objects they are close to, depending on the chosen modality. Both solutions, however, generate signals that are completely out of context.

In [10], authors describe an engagement-dependent AV, selectively incorporating aspects of reality only when needed, designed to preserve immersion and avoiding the frustration of removing an HMD. Such a system is a sort of open window into the real world, with different levels of blending into the virtual one. Again, the system add out of context information to the virtual environment.

The Point Cloud Method [6], instead, displays 3D point-cloud data in the virtual space where there are objects in the real world. Using point-cloud data allows the users to recognize the virtual world and real world information simultaneously, because they can see through the point-clouds. A camera is used to obtain the color and distance information of the real space. In this method, to provide spatial information of the real space, the point-cloud are placed in the virtual space based on depth information. To prevent a loss of immersive feeling in the virtual space, point-cloud data that does not have much contribution to understanding the real space are removed. Therefore, placing a point cloud that is over a predetermined distance

1https://support.steampowered.com/kb_article.php?ref=6281-TOKV-4722
2https://www.vive.com/us/
from the user is useless because the real-space information that are far away from the user’s position do not affect his movements significantly, nor allow physical interaction. Again, the presence of a point cloud out of the context of the virtual environment could negatively affect the sense of presence. It is worth noting that even different visualizations that are out of context, such as occupancy map and glass wall, do not improve the sense of presence [7, 9]. Other similar approaches [18, 23] add to the virtual environment layers, such as wireframes or other visual indicators, to show the presence of real obstacles.

In [19], authors proposed the idea of rebuilding a whole virtual environment based on the 3D model of the room where the VR system was set up. They detected the walkable area and inserted visual barriers, like fences or water, in the virtual world to define it. Moreover, specific objects are recognized and substituted in the virtual scene. The approach is promising for the purposes of avoiding collisions and allowing haptic interaction, but it is limited to specific objects.

Another recent approach, more focused on physical interaction than on obstacle avoidance, is described in [3]. Here, a mixed reality scenario is obtained by detecting the 3D position and orientation of a mannequin, and putting a virtual aligned copy of it inside VR. In this case, the level of presence in the virtual environment remain high, but the approach is not general, since the solution has been developed ad-hoc for the specific scenario. In [11] the authors propose to augment a virtual environment with real objects by embedding color and stereo information that are obtained by the stereo camera system of the HTC Vive Pro in order to allow a collaboration, however the latency and moderate resolution limit the proposed system.

Based on a non-visual feedback, in [22] the authors propose to embed vibro-tactile actuators into the HMD’s face cushion, to provide an alert when users are approaching obstacles.

In this paper, we overcome the limits of the existing approaches by devising a system which preserves the sense of presence and which is generalizable to most of the application contexts. Indeed, the virtual scene is created by taking into account the size and the position of the real-world obstacles, but no out-of-context information are displayed to the user. The method does not have specific priors nor on the real environment, where the HMD is used, or on the virtual scene.

2 Proposed method

The method proposed in the current work is divided in two main parts (Figure 1), registration and swap. Registration includes all the stages necessary to obtain a virtual scenario integrating information about the real environment surrounding the setup. First, it is necessary to reconstruct the 3D model of the real room. Then, some preliminary operations are applied to the model: floor detection; distinction of free and filled spaces, substitution of the 3D model single mesh with a modular mesh and recognition of the different elements occupying the playing area in the room model. In this case, we used simple clustering techniques, allowing to separate the occupied volume in different bulky elements. Finally, swap is the last phase, where virtual representations of real obstacles are substituted with items coherent with the VR scenario.

2.1 3D model of the environment

The first phase consists in the generation of a room reconstruction, the 3D model is created using the data captured by an RGB-D sensor. In our specific implementation we used a Microsoft Kinect and the Skanect software. The literature is rich in methods to obtain dense 3D reconstructions using both RGB and RGB-D sensors [4] [5] [8] [21]: the technique choice can thus vary depending on the specific settings of the scenario. Few adjustments to the computed 3D model may be done in a post-processing phase by using 3D computer graphic software, such as Blender.

2.2 Floor detection

Once the model is imported into Unity 3D, the floor of the reconstructed room has to be detected.

The previously obtained model is likely to include some degree of noise, like uneven surfaces (due to the sensor precision/reconstruction errors) or holes (caused by blind zones during the reconstruction process). Assuming that the biggest horizontal surfaces in a closed room mesh are the floor and the ceiling, to locate the floor, we slid an horizontal plane upwards through the model, until it reached half of its height (to discard the ceiling). With the exception of a few very specific ill-defined situations, the maximum intersection area is most likely found in the floor position (Figure 2). As Unity does not provide a method to calculate the

![Figure 1: Workflow of the proposed method, divided in registration phase (blue boxes) and actual swap (green part).](https://www.blender.org/

![Figure 2: To detect the floor, a planar square matrix is moved upwards through the model of the room: the maximum intersection area is generally in close proximity to the floor. The intersection area is obtained by computing the number of tiles of the matrix which collide with the model.](https://skanect.occipital.com/)
overlapping area of two colliding meshes, the intersecting plane has been implemented with a matrix of cubes. The floor is thus detected maximising the number of cubes of the matrix which collide with the model, while the matrix is sliding through the model. The accuracy of this Floor Detection method is clearly directly proportional to the matrix resolution.

After the floor position has been found, the model is shifted to make its base position coincide with the 0 value of the vertical y axis, and a horizontal plane is added to cover any potential holes in the mesh.

### 2.3 Scanning and rebuilding

At this stage of the workflow, the model of the room is still composed of a single mesh. Thus, the next phase consists in the creation of a modular mesh.

The method is similar to the one used for Floor Detection. A matrix of cubes is slid through the model and instantiates a virtual object every time a cube collides with the model. The matrix resolution is again proportionally tied to the accuracy of the reconstruction, and inversely proportional to execution time. Since creating a matrix of objects with high resolution can result in a high number of objects instantiated, it is possible to perform the scan by sliding a single row of cubes over two dimensions, or a single cube over three dimensions (Figure 3). In general, the three different solutions have a polynomial increment in performance (with the single cube performing the most operations, and the matrix the least). In our case, this phase is intended to be a pre-processing operation before the actual usage of the VR environment, thus we did not have particular efficiency constraints.

![Single Cube, Row, Matrix](image)

**Figure 3:** The three different solutions for the “scanning” method

Once collision are detected, the parts of the single mesh model can be rebuilt using two different approaches: the Voxelization and the Tiny-Voxelization method.

In the Voxelization method [2], when a voxel (represented by cubes in Figure 3) collides with the mesh, a copy of itself is instantiated in the scene. The obtained segmented scene is composed of cubes (Figure 5(b) top).

In the Tiny-Voxelization method, as soon as a voxel collides with the mesh, a pillar (a parallelepiped) is instantiated from the position of the collision to the floor (Figure 5(b) bottom). This solution reduces the accuracy of the reconstruction (by filling holes), but decreases the number of objects instantiated. The Tiny-Voxelization approach excludes the existence of floating objects. In the scenario proposed in this paper, floating objects were unwanted, but could be useful in other environments - especially in the perspective of improving augmented virtuality and interaction with the real space.

### 2.4 Clustering

In the Clustering phase, cubes or pillars instantiated in the previous step are grouped together into a new single object, which, in the best case, should correspond to a specific element in the real environment (Figure 5(c)). The clustering method implemented is based on closeness.

The closeness criteria has been implemented by defining a spherical volume which encapsulate a single voxel of the mesh, and merging into the same cluster all of its connected voxels. All the voxels of a cluster are thus either directly connected or reachable from all the other voxels of the cluster. All the clusters must contain a minimum amount of voxels in order to be registered (in the current implementation this minimum value has been set to 10 voxels).

Ideally, the volumes obtained by the clustering should approximate the mesh while maintaining as much detail as possible. When applying the clustering algorithm on the mesh segmented with the Tiny-Voxelization, it is possible to also perform a check based on the height of the neighbor voxels, to increase the fidelity of the reconstruction (Figure 4(d-e)). The "check on the height" consists in grouping together the volumes from the Tiny-Voxelization only if they have similar height (in the current implementation the threshold on the height is set to 0.1 units).

It is also desirable to make the virtual object volume as similar as possible as the volume of the cluster. In other words, we would like to put in the VR scene a virtual object which resembles the real one, in terms of shape and dimension. A sufficient condition to enable real world obstacle avoidance, is to put a virtual object which entirely includes the corresponding real one. In order to approximate by excess, the system replaces a cluster with the smallest parallelepiped containing all the objects belonging to that cluster.

### 2.5 Alignment of real and virtual world

After having created a virtual environment, taking into consideration real objects and obstacles present in the real surroundings, it is necessary to align the virtual and the real worlds. This process is needed both to effectively avoid obstacles in the real scene and to possibly interact with the real objects, in an Augmented Virtuality context.

The alignment is done by translating, rotating and scaling the virtual world to match the real one. All the transformations can be obtained by matching three 3D points with known position in both the virtual and real worlds. Since the floors have already been aligned, in our case matching two points is enough to obtain the remaining transformations. We decided to use as keypoints the positions of the two base stations of the HTC Vive, since they are both easily identifiable in the mesh and tracked in the VR system.

The scale factor $s$ applied to the mesh is obtained by computing the ratio of the distance between the two keypoints positions in the two different reference systems. Assuming $d_{base}$ is the distance between the two base stations as tracked by the HTC Vive system, and $d_{measure}$ is the same distance as measured from the mesh of the room, including the mesh of the two base stations, the scale factor is thus $s = d_{base}/d_{measure}$.

The translation transformation can be obtained by finding the relative position of the point in the center of the line connecting the two base stations in the mesh, with respect to the same point in the HTC Vive reference system. Once the origins of the reference systems are aligned, only one rotation needs to be found - the rotation which aligns the lines passing through the base stations in the two different reference systems. Since the lines are crossed (symmetric) with respect to the origin, it is sufficient to find the angle $\alpha$ which brings the vector $v_{htc}$ which points at one of the base stations in the HTC reference system, to the vector $v_{mesh}$ which points to the same base station in the mesh (Eq. 1).

$$\alpha = \arccos \left( \frac{v_{htc} \cdot v_{mesh}}{|v_{htc}| \cdot |v_{mesh}|} \right)$$

The axis of rotation on which $\alpha$ needs to be applied to achieve the full alignment is given by the cross product between the two vectors $v_{htc}$ and $v_{mesh}$.

### 2.6 Swapping

The last step of the system is the “swapping” one, with the final result of a virtual environment ready to use. Here the goal,
fact, is to replace the clusters with similar objects and fuse the reconstruction with a prebuilt background. In the future, we aim at building a dictionary of possible objects, contextualized in the specific virtual scene. Ideally, different environments correspond to specific dictionaries. Each element of these collections should be composed of a model and a descriptor, used to choose which item of the dictionary should replace a specific cluster with specific properties. As discussed before, as we base the swap action on the shape of the cluster without taking into account the actual replaced object shape, our descriptor contains three numbers representing the scales of the model along the three axes. Moreover, being the current work a demonstration of the possibilities the method offers, the current dictionary contains only six entries, corresponding to rocks to be inserted in a urban park virtual scenario. To find the replacing object we minimized the sum of the differences between the sizes of the object to replace and every entry in our dictionary.

\[
\text{Diff} = \min_{\forall e \in D} ((c_x/c_y) - (e_x/e_y))^2 + ((c_z/c_y) - (e_x/e_y))^2
\]  

(2)

Where \(e\) is an entry in the dictionary \(D\) and \(c\) is a cluster, \((c_x, c_y, c_z)\) are the sizes of the cluster (i.e. the size of the bounding box to be replaced) and \((e_x, e_y, e_z)\) are the sizes of the \(e_{ib}\) entry of the dictionary.

Finally, the best matching object in the dictionary is instantiated at the right position and, after normalizing its sizes, its scale is set as the same of the cluster (see Figure 6(a)).

The reconstruction obtained using the height clustering method is better in terms of error, but makes the scene more chaotic and often instantiated objects merge themselves, thus creating unnatural conglomerates (Figure 6(b)).

An overview of the 3D mesh of the room, overlapped with the virtual matching objects instantiated at the same positions of the real obstacles, is shown in Figure 7.

2.7 Examples with different environments

The previously described steps, i.e. the 3D Reconstruction of the scene, the Voxelization (or the Tiny-Voxelization), the clustering and the swapping, have been tested on several real-world environments. Figure 8 shows some examples taken from different contexts. In particular, a “kitchen”, a “relax corner” with a sofa, and a “countryside living room” have been considered. In all the considered scenes, the proposed method is able to build a VR scene with virtual
The resolution of the voxelization can be adapted with respect the desired accuracy of the final reconstruction (see Tab. 1 and 2).

3 EXPERIMENTAL VALIDATION

In this section, we analyze the developed method by considering: (i) the performance in terms of memory occupancy and execution time, by varying the desired resolution of the scene rebuilding; (ii) the accuracy of the object boundaries detection; and (iii) a preliminary user experience analysis, to understand whether the method produces an acceptable level of presence in VR.

3.1 Quantitative assessment

Table 1 shows the execution time in seconds, by considering the number of voxels ideally covering the entire 3D space (in the table we indicate the number of voxels per side of the scanned volume) and the 3 different scanning methods. The method has been tested on a Intel Core i7-6700K CPU at 4.00GHz. In particular, we propose results obtained with the Voxelization approaches: the single-voxel scanning voxelization (SC-Voxel), the row scanning voxelization (R-Voxel), the matrix scanning voxelization (M-Voxel), also represented in Figure 3, and finally with the Tiny-Voxelization and matrix scanning (M-tinyVoxel). We refer to a model of a real room whose size in metres is $4.53 \times 3.69 \times 5.08$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>445</td>
<td>15</td>
<td>7.8</td>
<td>7.6</td>
</tr>
<tr>
<td>60</td>
<td>3721</td>
<td>63</td>
<td>9.7</td>
<td>7.8</td>
</tr>
<tr>
<td>90</td>
<td>12859</td>
<td>145</td>
<td>13.1</td>
<td>11.9</td>
</tr>
<tr>
<td>120</td>
<td>30310</td>
<td>257</td>
<td>23.6</td>
<td>21.0</td>
</tr>
<tr>
<td>150</td>
<td>53623</td>
<td>411</td>
<td>37.6</td>
<td>34.3</td>
</tr>
</tbody>
</table>

objects (in our case rocks) replacing the real obstacles in the scene. The first step is the reconstruction of the floor (visible in the middle subfigures of each panel). This step does not need an accurate scan of the real floor (see as an example the scan of the “countyside living room” scene). In general, the objects in the scene are detected and their occupancy correctly estimated, even when the object has a complex structure (see the red chairs in the “countyside living room” scene, or the table in the “relax corner” scene, whose occupancy volumes are highlighted in orange in the corresponding top-right images). It is worth noting that sometimes different objects collapse into one occupancy volume (see for example the trash and the paper roll, or the sink and the hob in the “kitchen” scene highlighted in orange in Figure 8 (top-right)).
Figure 8: The proposed method applied to three real-world cluttered scenarios. Each panel shows a snapshot of the real environment (top-left), the 3D reconstruction of the scene (bottom-left), the segmentation of the mesh by Voxelization (top-middle) and Tiny-Voxelization (bottom-middle), the clustering to obtain the occupancy volumes to be swapped in the VR scene (top-right) and the obtained VR scene (bottom-right). The orange boxes in the (top-right) images highlight the clusterisation of complex objects such as the kitchen sink and hob, the table and the red chairs (see text for details).
The execution time has a quadratic growth for the single-voxel scanning Voxelizeation, a linear increase in the case of the row scanning Voxelizeation and it remains almost constant for the matrix scanning Voxelizeation. These results are affected by the limited number of voxels in height. The Voxelizeation and the Tiny-Voxelization approaches have not an influence on the execution time.

Table 2 shows the variation in the number of instantiated objects by varying the number of voxels per side and the rebuilding method.

Table 2: Number of instantiated objects with the two rebuilding techniques, with respect to the resolution of the scanning matrix

<table>
<thead>
<tr>
<th>Voxel/side</th>
<th>Voxelizeation</th>
<th>Tiny-Voxelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2211</td>
<td>473</td>
</tr>
<tr>
<td>60</td>
<td>9749</td>
<td>1654</td>
</tr>
<tr>
<td>90</td>
<td>22922</td>
<td>3564</td>
</tr>
<tr>
<td>120</td>
<td>41138</td>
<td>6123</td>
</tr>
<tr>
<td>150</td>
<td>65716</td>
<td>9423</td>
</tr>
</tbody>
</table>

It is worth noting that the number of instantiated elements depends on the actual presence of objects in the scene and on their size. In any case, the Tiny-Voxelization technique is less demanding in terms of memory occupancy.

Since the main goal of this paper is to present a flexible method to detect objects in a real scene and introduce them in a virtual environment, we now decide to focus on the Tiny-Voxelization technique with the matrix scanning, which is the best choice in terms of execution time and memory occupancy. Figures 9 and 4 show the result of the reconstruction for two specific objects in a room, one characterized by a simple shape and the other having a more complex structure. Considering the Tiny-Voxelization, it can be noticed how the final result strongly depends on the clustering method used, especially in the case of the complex element (Figure 4). In particular, the structure composed of the monitor and the table is represented by a single cluster, if the “clustering by closeness” technique is used, and by two different boxes, with the “clustering by height” variant.

Table 2 shows the variation in the number of instantiated objects by varying the number of voxels per side and the rebuilding method.

Table 2: Number of instantiated objects with the two rebuilding techniques, with respect to the resolution of the scanning matrix

<table>
<thead>
<tr>
<th>Voxel/side</th>
<th>Voxelizeation</th>
<th>Tiny-Voxelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2211</td>
<td>473</td>
</tr>
<tr>
<td>60</td>
<td>9749</td>
<td>1654</td>
</tr>
<tr>
<td>90</td>
<td>22922</td>
<td>3564</td>
</tr>
<tr>
<td>120</td>
<td>41138</td>
<td>6123</td>
</tr>
<tr>
<td>150</td>
<td>65716</td>
<td>9423</td>
</tr>
</tbody>
</table>

The choice of the clustering techniques should be further influenced by the final goal of the system: obstacle avoidance would require a safer approach (thus virtual objects bigger than the real ones), whereas interaction should have benefits from a more accurate swap.

3.2 User experience validation

We performed a preliminary user experience experiment, to assess if the solutions we implemented can produce an appreciable variation of immersivity and sense of presence with respect to the state of the art Chaperone method. In particular, we compared the Chaperone with the scene obtained combining the Tiny-Voxelization with the clustering by closeness and clustering by height approach, which have been previously proven to be the most efficient solutions, in terms of execution time and number of instantiated objects. Nevertheless, we expect to find a noticeable increase of sense of presence in the clustering by height condition, as it allows to reproduce items in the real world in a more detailed way.

For the three trials, we implemented a virtual scenario representing a wood. Obstacles of the real scene were replaced with stones, as described in the previous Section.

Sense of presence was measured using the Igroup Presence Questionnaire (IPQ), which is one of the standard questionnaire currently available for measuring presence in a virtual environment (VE) [12, 13]. The questionnaire is composed of 13 7-points Likert scale questions which evaluates three main aspects related to sense of presence: the Spatial Presence, the sense of being “physically there” in the virtual environment; the Involvement, considered both as attention during the interaction and as perceived involvement; the Experienced Realism of the VR experience. An additional question rates the sense of presence from the original definition on [17].

Figure 10 shows the average values of the IPQ parameters for the three solutions we compared.
Six participants, aged between twenty-three and twenty-nine years with an average of twenty-five, all novel to virtual reality or with low experience in the matter, took part to the experimental session. They all had normal or corrected to normal vision. The experiment follows a Within-Subject design: each subject tested the 3 conditions in randomized order. Participants were instructed to freely explore the scene, especially getting close to the obstacles added in the virtual environment or to the boundaries of the tracked area. They had a HTC Vive controller and they were instructed to "touch" the obstacles with the controller. The total permanence in the VE was about 5 minutes. After each exposure subjects were asked to fill the IPQ questionnaire after every session.

Answers show an higher level of sense of presence in the implemented methods than in the standard one. Moreover, especially in the trials obtained using our method, users felt aware and confident of the position of real obstacles in the room (see Figure 11, representing a specific snapshot, where a user is told "as soon as you see a small stone on the floor, reach it and put the controller on it") and in the online video. The preliminary results we obtained are promising, and they show the the proposed system has the potential to be applied for creating a safe space for the interaction in a VE through augmented virtuality. Further experimental evaluation is necessary in order to assess both the sense of presence [15] and the interaction.

4 Conclusion

In this paper, we propose a method to improve the user’s awareness about the 3D structure of the real world, when she/he is acting in immersive virtual reality. The aim is to create a virtual environment that takes into consideration the presence of the real obstacles, by adding to the virtual environment contextualized virtual objects having the same position and occupancy of the detected real ones. In this way, the user can avoid the collision with the real objects, and eventually interact with them.

We implemented a method that performs a voxelization and clustering of the acquired data of the 3D scene structure through a RGB-D device in order to obtain 3D models of scene objects. Then, such 3D models are swapped with corresponding contextualized virtual objects.

We devised different techniques for the scanning and voxelization of the scene and analyzed them in terms of memory occupancy and time of execution: such an analysis allowed us to find a flexible technique to detect objects in a real scene and embed them in a virtual environment (i.e. Tiny-Voxelization technique with the matrix scanning). Then, we considered the accuracy of the reconstruction for the scene objects: we compared the two proposed clustering techniques in order to quantify the error between the occupancy of the real objects in the scene and in its virtual counterpart. The results showed that we can choose the accuracy as a function of the task: for obstacle avoidance that requires a safer approach the system is able to produce objects bigger than the real ones, whereas for interaction we can get a more accurate swap (i.e. to replace the clusters with similar objects).

The method has been successfully applied to several real world scenarios, having different structural complexities (e.g. objects on the floor, big furniture, small details), thus showing the possibility of being used in any indoor real context. Finally, a pilot user experience study showed that the proposed system allows a higher level of sense of presence than the standard Chaperone technique. Thus, such results suggest that our system produces a safe space for the interaction in a VE through augmented virtuality.

The main limitation of the devised method is that it relies on an offline scan of the real scene, which cannot be obtained in real-time, yet. Nevertheless, the possibility of tracking in real-time the clusterized objects will allow an effective use of the proposed technique in immersive VR applications. Indeed, once the scene is reconstructed, any change (e.g. the position of objects) can be detected and updated into the VR scene.

Moreover, the system could be improved by taking into consideration several VR scenarios and dictionaries also embedding a semantic description of the virtual objects to be swapped. In this way, virtual objects could be inserted also by taking into consideration features like their common use and position in real environments. Finally, more experiments to assess the quality of interaction in the augmented virtuality scenario are necessary, to further validate the technique.

Figure 11: An user is asked to touch the black stone he sees on the floor (bottom), and he really touch an obstacle in a room having the same position and size (top).
REFERENCES