Reconstruction of Wall Surfaces Under Occlusion and Clutter in 3D Indoor Environments

Antonio Adán\textsuperscript{1} \hspace{1cm} Daniel Huber\textsuperscript{2}

CMU-RI-TR-10-12

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

April 2010

\textsuperscript{1}Castilla La Mancha University, Ciudad Real, SPAIN
\textsuperscript{2}Carnegie Mellon University, Pittsburgh, PA, USA
Abstract

In this paper, we present a method for the reconstruction of interiors using a set of panoramic range data in scenes with clutter and occlusion. We specifically deal with the reconstruction of simply-shaped wide areas (such as walls, ceilings and floors) behind furniture and facility pieces in interiors. To date, little attention has been paid to this issue and only incomplete solutions in simple scenarios appear in literature. This document presents an integrated solution to this problem, ranging from the data collection to the restoration of missing 3D information. Our approach is based on a sequential updating labeling strategy in different data representation spaces. A volumetric representation is used to permit the labeling of the 3D space for different range data and the fusion of all the scene’s labels to obtain one single 2D labeling image for each of the simply-shaped wide areas. Based on this labeling process, our method is able both to identify occluded regions and, through an SVM learning technique, to recognize essential parts of the walls, such as doors and windows, so that labeling is continuously updated. Finally, the reconstruction of the wall is carried out in the last stage of the process by using an inpainting algorithm, which has been adapted to our particular application. The method was tested in real generic scenes under difficult clutter and occlusion conditions, and has yielded promising results.
1. Introduction

Range images are used in many applications related to a variety of environments. Apart from object modeling and synthesizing, manufacturing and reverse engineering, computer vision and robotic context, 3D data recovery technologies are also extensively applied in the areas of civil engineering and architecture. Over the last few decades, laser sensors have become more accurate and adaptable to work both inside and outside buildings. This technology has, therefore, been extensively used for large-scale modeling and building quality evaluation.

Obtaining and representing an accurate 3D model as-built condition is an enormously time-consuming process that is normally carried out manually. Nevertheless, automated solutions to obtain building information models (BIM) are currently required. Note that an as-built model of a facility might, for various reasons, differ from the initial design of the building during construction. An as-built BIM is therefore a necessary tool to aid in further maintenance and structural changes of the building.

The content of this document is framed in the BIM context. The first goal of this work is to automatically identify essential parts of a building by using range data under occlusion circumstances. Essential parts are walls, ceiling and floors. Although this objective might seem simple when the building is uninhabited, it becomes more difficult when dealing with inhabited spaces containing a wide variety of objects, such as chairs, tables, lights, paintings, notice boards, panels, and so on. Figure 1 shows an example of a complex scene in which our method is being applied. This kind of environment leads to clutter and occlusion problems. It is consequently necessary to seek new solutions that will identify which parts of the walls are occluded by objects located in front of the sensors. This problem is solved by defining a voxelized representation of the scene and performing a voxel labeling fusion from different sensor positions, as will be shown in Section 4.

The second objective of this work has two aims: firstly, to recognize which parts of the occluded zones belong to the wall and which do not; and secondly, to apply an efficient gap filling algorithm that totally reconstructs the wall. In summary, the subject of this paper can be stated as the detection and reconstruction of essential parts of a building under clutter and occlusion circumstances using a set of range images.

Figure 1. A complex interior scene with multiple occlusions and clutter.

The paper is organized as follows. In Section 2, a discussion of the most relevant work concerning occlusion detection in range data and 3D reconstruction in large environments is presented. Section 3 shows an overview of our solution. Section 4 is devoted to the labeling procedure, including the following stages: 3D voxel representation, wall identification, voxel labeling, and 3D data labeling. Section 5.1 introduces the problem of detecting occluded regions as a requirement for carrying out the reconstruction of the wall. Feature extraction in the wall and recognition algorithm of what we shall denominate as holes are explained in Sections 5.2 and 5.3. The filling algorithm is briefly tackled in Section 6. Section 7 presents our
experimental results, and the paper concludes in Section 8, which sets out its main contributions and future improvements.

2. Previous Work and Contributions of Our Method

The occlusion problem appears in several areas related to vision from different perspectives. For example, in the field of 3D visualization and graphics, the occlusion effect has a detrimental impact on tasks involving discovery, access, and spatial relation of objects in the scene. Elmqvist et al. [Elmqvist 2008] define a taxonomy of 3D occlusion, formalizing a common terminology and theoretical framework. In the field of computer vision, occlusion problems arise in a wide variety of applications related to recognition, image analysis, image understanding [Hoiem 2008], data registration, scene reconstruction, autonomous robots, vision-based robot manipulation, etc. On the other hand, in 3D reconstruction works using LiDAR range data, occlusion in the scene is considered as one of the principal sources of uncertainty. If this issue is to be dealt with effectively, a 3D object detection strategy fusing data taken from different viewpoints and under an occlusion model must be carried out.

To date, few papers have tackled the occlusion problem by using range data for indoor reconstruction. Of course, a multitude of related papers exist that use range images and deal with specific topics such as plane detection, wall identification, 3D scene segmentation, reconstruction of surfaces, punctual occlusion and hole-filling, but none of them provide an integrated solution in complex scenes covering all those aspects. A review of current techniques concerning occlusion and reconstruction aspects using range images for indoor and outdoor environments follows.

Reconstruction of planar surfaces under occlusion has been tackled by F. Dell’Acqua et. al [Dell 2002]. This paper presents simple scenarios in which a single object occludes a planar surface. The method is based on finding depth discontinuities and segmentation in the depth image. A region reconnection stage is then carried out to determine the possible occluded regions in the background plane. The authors investigate the area between the boundaries of the regions. An occlusion condition is first imposed over the pixels by finding occluding lines between all possible pairs of boundary-pixels in two segmented regions. Two regions are finally considered to be separated by an occluded object if the number of occluded lines exceeds a certain quantity. The gaps in the occluded area are filled through the assignation of a new depth by interpolating the depth of the end points of the shortest occluded line. In [Stulp 2001] this work is extended to cylindrical and spherical surfaces in very simple cases and always using a single range image.

Most of the occluded region detection solutions appear after a range segmentation stage. For instance, Sappa presents an approach that recovers occluded regions in scenes with planes and cylinders [Sappa 2002]. He assumes that the segmented model of the scene is provided in advance. A region connectivity graph between regions and using the distances from the regions to the sensor is used to identify occluded regions and niches. When the gaps correspond to a cylinder, a parametric fitting function is applied to reconstruct it. If the occluded region is in a plane shape, he defines an iterative algorithm which calculates lines between the two planar segments until the gaps are filled. The work is oriented towards studying the continuity of surfaces and the automatic generation of surface models. Two examples in a reduced scenario are shown in the report presented in [Sappa 2002]. The approach presented by F. Han et. al in [Han 2002] is devoted to segmenting a range image into a number of surfaces that are further
classified. The authors achieve optimal segmentation by using Markov chains in a Bayesian statistical framework. In this work, panoramic indoor and outdoor range images are used. Nevertheless, although the authors recognize that little attention has been paid to the reconstruction of occluded parts in this kind of complex environment to date, they do not properly deal with the occlusion and filling problem, considering it as a future work.

Occlusion and scene reconstruction solutions using range images in lab environments have also appeared over the last few years. In this context, occlusions are caused by the objects of which the scene is composed. The approach presented by U. Castellani et. al. in [Castellani 2002] is focused on the reconstruction of polyhedral objects in scenes with occlusion. These authors specifically tackle the reconstruction of corners and edges of partially occluded simple-shaped objects. A segmentation of the scene to extract true and false boundaries of each object is first carried out. The boundaries are then labeled based on the relation between visible boundaries, and the continuation of incomplete boundaries is eventually estimated. After completing the boundary reconstruction, hypothetical surfaces can be created by extending the visible surface regions into the identified bounded area. In this environment, many other reconstructions solutions use 3D recognition techniques. In this case, the objects of the scene are recognized and their poses are calculated [John 1999], [Mian 2006], [Merc 2007]. The 3D models of the objects can then be placed in the scene and the scene is reconstructed. The occlusion regions are not, therefore, explicitly extracted. Such techniques could obviously be related to the occlusion problem in range images for indoor scenarios if the 3D models of all the objects in the scene were stored in advance. Nevertheless, this is unlikely owing to the huge variety of objects that might exist in a real scene.

Automatic 3D restoration is also tackled from a set of 2D images. In this case, occlusion detection becomes essential to the generation of the whole 3D model. The occlusion can cause several problems, particularly the many false 2D points that appear when using edge detectors, consequently leading to the incorrectness of all further correspondences to integrated images which contain those points. Another problem concerns the detection of the occlusion and finding the borders of the obscured parts. In [Pospisilova 2007], the authors focus on full automatic completion of the gaps in 3D building models. They identify the covered parts that are not visible in any image and reconstruct the gaps by means of a model fitting approach. In this work, the models of several objects and certain properties of the models (shape, color, texture, etc.) are known in advance. Small invisible parts of the surface are thus found by fitting the model into the scene. This is a useful strategy for the reconstruction of repeated models in the scene. [Dick 2001] and [Werner 2002] deal with a building that contains windows of the same shape. In this case, the model of the best displayed windows can be chosen as the pattern and substituted for another distorted or occluded window. A similar work is developed in [Böhm 2007] in which a semi-automated repair of facades is tackled. One limitation in this work is that a user must mark the defective area in a range image. Assuming that most defects are caused by incomplete data, a further improvement is made by identifying areas which have insufficient sampling. The authors then automatically search for a similar area by using a simple template matching. The method is tested in a very simplified scenario consisting of a facade with repetitive windows in which occlusion circumstances are discarded [Böhm 2008], [Beck 2007].

View-planning for automated data acquisition in large indoor and outdoor sites is frequently related to occlusion problems. For 3D reconstruction of the scene, several scans are taken, the first of which is carried out by a human operator. Nevertheless, the subsequent scans are taken in positions according to a planner, so that the occlusion is minimized as new scans are taken.
The principal problem then consists of choosing the next best view (NBV) of the scanner. As will be shown in the following section, the technique presented in this paper is related to the NBV works in two ways. Firstly the NBV planning is tackled through the introduction of a 3D representation model of the space, usually voxels [Sanchiz 1999]. In this case, we can define a voxelization of the space and carry out a voxel classification according to the visibility criteria from the sensor position. The second common point is that all NBV approaches deal with multiple views as in our proposal. Of course, the sequential search of the NBV does not make sense in our case, where all the range images are provided in advance. Nevertheless, an integration of 3D information takes place in both cases. The most interesting approaches in this area are as follows.

Blaer et al. in [Blaer 2006] and [Blaer 2007] present an automated data acquisition in two phases. An initial model of the target region is calculated in the first stage using a two dimensional map of the region. The planning of these locations resembles the art gallery problem. Thus, they scan from a set of positions to allow all the walls in the environment to be imaged. A second stage refines this initial model by providing additional 3D information lacking in the first version. In this stage, a voxel representation of the data is introduced by labeling voxel as unseen, seen-empty, or seen occupied. The position that maximizes the number of unseen voxels is then chosen as the NBV. The authors describe the occluded regions when they search for voxels that fall on the boundaries between seen-empty regions and unseen regions [Blaer 06-2]. These boundary regions are most likely to contain occluded zones. As the authors themselves admit, the requirement of a 2D map is too restrictive. The method has been tested in large indoor and outdoor structures. Low et al. [Low 2006] [Low 2006-2] tackle the NBV problem in range sensing directly, but introduce space properties that are connected with occlusion aspects. They formulate a general view optimization metric which is able to include most of the acquisition and reconstruction requirements. The surfaces of the volumes are classified in a partial model of the single scan as: true, false, occlusion, hole-boundary and image-boundary. A hierarchical evaluation of the NBV is accomplished through a cost function which, apart from acquisition constraints, considers the weight or importance assigned to the surface type in the current scene model. For example, occlusion surfaces will be assigned to large values whereas true surfaces will have small values. If all the views are scored below a specific threshold, the view planner will suggest the termination of the acquisition process. The whole technique is implemented by using both octree and mesh models, which leads to a thorny adaptation from one model to another when a new scan appears. The system has been tested on simulated and real scenes.

Klein et. al. [Klein 2000] address the problem of view planning with an initially unknown geometry of the scene, and present a surface representation of seen and unseen parts. They propose an objective function based on the analysis of occlusions, but by incorporating a quality criterion. The goal is to achieve a complete surface model measured with a predefined sampling density in each point. Thus, a cost function takes into account diverse information: visibility, 3D position, surface normal and the quality measure. Again the occluded space is not explicitly calculated. In an earlier work [Mass 1998], Massios and Fisher also defined a quality criterion in addition to the usual criterion. They use a volumetric representation called voxelmap in which each voxel is an identical cube that is labeled as empty, seen, or unseen. They define the new concept occlusion plane as being the plane in which an unseen voxel is the neighbor of one or more empty voxels. Furthermore, a quality value is assigned to seen voxels and a region quality measure can eventually be obtained. These criteria are used for NBV evaluation: first, by maximizing the amount of occlusion plane voxels and second, by maximizing the amount of
low quality voxels that are visible from the new view. The method is tested in a reduced setup with small objects so that its utility in large spaces is questionable. A probabilistic representation of free, occupied and hidden space is presented by Yapo et al. in [Yapo 2008]. These authors propose a discrete scene data structure in which a probability that any single 3D voxel is in one of the three states is defined. The experimentation is reduced to obtain detection of two single objects partially occluded by a camouflage net. Finally [Sanchiz 1999] defines an effective and efficient algorithm for scanning an environment such that all surfaces are observed with high quality measurements. The sensed quality of an occupied voxel is the cosine of the angle formed by the surface normal and the viewing ray. The voxel quality is the best sensed quality of the voxel until that moment. The NBV is chosen by optimizing a quality function over the newly occupied voxels. The goodness of the method was experimented on in a simulated range sensor and mobile base.

Detection and hole-filling techniques are also related to occlusion problems in the 3D reconstruction for large environments. Despite the fact that a huge quantity of hole-filling algorithms are applied in 3D object restoration, most of them are focused on extracting an accurate reconstruction in old heritage pieces, and few methods in the building reconstruction context can be found in literature. According to current laser sensor outputs, most of these methods use mesh representations to discover inconsistency or discontinuities in the sensed surface. This kind of representation is hardly ever useful in a large environment because of the unapproachable complexity and computational cost that it entails.

There are two reasons for the appearance of holes in a mesh. One is that of an incorrect redefinition or loss of data in the registration and/or integration process. This kind of hole does not involve a difficult problem and is filled without considering the surface information of the surroundings. The second reason is geometric information loss during the acquisition process. This may be because the sensor does not provide enough density of points of the surface or because the laser is unable to reach the surface owing to occlusion circumstances. In general two strategies are considered to solve the lack of 3D information. There are methods which deal with the hole-filling as an implicit task to be solved in the generation of the 3D model representation and methods in which it is a post process of the 3D data acquisition.

The problem of filling-holes in range data can be divided into two stages: identifying the holes and finding appropriate parameterizations of the gaps that allow reconstruction.

As was mentioned previously, few automatic hole-filling solutions have been developed on large geometrical structures corresponding to indoor scenarios (such as walls or floors) and in complex scene contexts. Most of the methods are applied over one single object mesh model as an isolated item. One of the papers dealing with filling in large spaces corresponds to reference [Wang 2002]. In this, Wang et al. tackle the hole-filling problem on symmetry objects in an indoor scenario. A reconstruction of a textured scene through several range images is carried out by following a pipeline which includes segmentation and reconstruction of planar surfaces, texture reconstruction, object clustering, symmetry checking, and surface fitting. In this last stage, an object that presents holes in its surface is automatically restored, taking into account the symmetry property of the object (for instance in chairs or computer monitors). In other cases without symmetry, the texture is reconstructed using a close-brush tool and a pull-push algorithm. In [Wang 2003] the authors present the algorithm to be run on smooth surfaces and do not therefore provide a general solution. After taking a mesh representation, the boundary of the hole is obtained and, after placing a vicinity ring around it, they use a moving least square interpolation approach. A recent solution to the restoration of hidden or missing portions of
objects via non-parametric techniques is presented in [Breckon 08]. The method is based on the non-parametric propagation of the available scene knowledge from the unknown area. These authors consider two stages: 1) calculation of the geometric completion of a smooth underlying surface model and 2) propagation of a part of the known surface over the geometric completion. This technique is applied on 3D repetitive pattern shapes and is limited to plane, cylindrical and spherical surface models.

In comparison with other previous work related to 3D reconstruction under occlusion and clutter, this paper makes the following contributions. Firstly, we apply our method in panoramic scenes which are frequently more complex than most of the lab and reduced scenarios used by others. Natural scenes like ours contain a huge quantity of unknown objects and elements that occlude walls, ceiling and floor. Objects placed on the whiteboards or blinds may even appear in our environments. This makes our method more challenging than others.

Secondly, we integrate several range images in a single framework to accomplish the identification of occluded regions in walls. Furthermore, we formally define a new voxel map representation and 3D label fusion which is finally adapted and refined for walls, this stage being the structural basis of the rest of the algorithm. Until now, the integration of several views has frequently been accomplished in view planning and NBV contexts but not in the reconstruction of large flat surfaces behind occlusions in range images.

Thirdly, we incorporate the hole object concept as a generic concept rather than using the specific kinds of objects (such as doors or windows) frequently used by others authors. This new concept allows us to tackle the hole-filling problem in walls in a generic and extensive manner, and to identify all the parts of the wall that should not be filled for reconstruction. Another aspect which should be mentioned is that we have designed a new strategy with which to recognize these holes by means of supervised learning techniques. Specifically, we have created a binary classification, using a Super Vector Machine algorithm, by defining a feature vector which integrates geometric and labeling characteristics.

3. Overview of the Method

In this section we briefly present our wall reconstruction strategy, detailing the main stages of the process. The general procedure can be split into five parts: data preprocessing, wall detection and voxel labeling, label fusion, hole recognition, and hole-filling. Figure 2 shows an outline of the process, detailing the principal steps in each stage.

We first filter outliers and erroneous data, and we then merge multiple range images into a single range image of the scene to obtain a 3D common data space, denoted as \( \Gamma \) in Figure 2. In the second stage, a volumetric representation is taken. The voxel space will be denoted as \( \Omega \). A new strategy is then introduced into this space to extract the walls of the scene and to classify the voxels according the occlusion circumstances. The voxels in \( \Omega \) are thus labeled as empty (A), occupied (B) and occluded (C) from each of the range images corresponding to different laser scanner positions. In this work, we pay particular attention to the walls’ voxel labels, and define the voxels that the surface of the wall contains as Zone 0. The remaining stages of the process are followed for each detected wall.

The third stage consists of integrating all labeled Zones 0 into a single representation and mapping this information onto the original 3D data to generate a high resolution labeled-projected-image of the wall. This image appears as image \( I \) in Figure 2. Image \( I \) can therefore
be viewed as a grid in which a pixel represents a small portion of the wall’s surface to which one label is associated. At this point, a new non-sensed (D) label might appear in sparse data areas.

In order to discover which parts of the wall must be filled and which must not, a segmentation of occluded regions is carried out in the fourth phase. In this phase, a new strategy is introduced to recognize what we have denominated as ‘holes.’ A hole in a wall is a region which is not a part of the wall structure (e.g., doorways and windows). This has been done by using a supervised learning approach based on the SVM technique [Vapnik 1998]. As a result of this, image \( I \) is re-labeled according our wall reconstruction goal. Thus, we finally define the hole, occupied and to be filled labels.

Stage five deals with filling the wall’s occluded parts (corresponding with the to be filled labels) that have been extracted in the previous step. This is done by running an image inpainting algorithm [Perez 2008] over the depth image \( \delta \) which is dual to \( I \). Once the image has been repaired, the inverse 2D to 3D transformation is performed and the reconstructed wall is obtained.

In order to make the whole process more understandable, Figure 2 contains three colors to show what kind of representation each step consists of. As we can see, there are three representation spaces throughout the whole process: a 3D data representation, a voxel representation, and a 2D representation.

![Figure 2. Outline of the wall reconstruction process.](image)

### 4. Labeling on Walls

In this section we introduce a 3D voxel representation of the space as a structured framework in which the majority of the algorithms will be developed. In general, the procedure will consist of labeling the voxel-space and later transferring the results to the original 3D data. The first stage of the process concerns the wall identification problem.

#### 4.1 Wall identification in a 3D voxel representation
As was previously mentioned, we are dealing with a set of range images which have been taken from different positions in a room. Although reflectance values are available, only the 3D data is used in our approach. After registering 3D data from different positions of the laser scanner, we can recover all available geometric information from the scene: a multitude of unconnected 3D points have to be processed. As will be shown later, apart from outliers and noise, the point density is usually irregular having overcrowded and low density areas on the walls. The volume of the room is manually selected, and 3D data corresponding to other rooms or outside space is removed.

In order to define an environment with an imposed topology over the data, we have voxelized the workspace in a multitude of minute cubes. We thus create a reduced and discrete tridimensional space in which it is possible to assign a label to each coordinate \((x, y, z)\). A point belonging to the space can, from the outset, be labeled as \textit{empty} or \textit{occupied}, and is considered to be \textit{occupied} when the assigned voxel contains at least one original point. In order to provide a coherent nomenclature throughout the paper, we will denote, on the one hand, the 3D data-space as \(\Gamma\) and \(p\) as a point in \(\Gamma\), and on the other hand, the symbol \(\Omega\) will be the voxel-space, \(q\) being a point of \(\Omega\). Thus, the scenario in a voxel representation can be seen as a discrete space with a six-connectivity topology. This structure will be very useful for representing and characterizing regions in the scene. Figure 3 shows an example of 3D data after integrating range data from five laser scanner positions, the demarcated space \(\Gamma\), the voxel-space \(\Omega\), and the set of \textit{occupied} points in \(\Omega\).

The use of voxel-space makes it is easy to determine the most important plane regions in the scene. For the time being, our goal is to extract the voxels that belong to walls, ceilings, and floors from the space \(\Omega\).

Automatic extraction of planar regions has been accomplished by several authors. One of the solutions applied to environments similar to ours can be found in [Sappa 2004]. The authors present a segmentation technique that decomposes a panoramic range image into a set of planar projections. They use two orthogonal surface orientation histograms and extract the principal directions. A histogram of distances is also computed in order to define the position of the projection planes in the world reference system. The original range image is then divided into as many planar projections as principal directions in the orientation histograms are extracted.
Previous to this work is that presented in [Huang 00] in which panoramic range images of forest scenes are dealt with. A brief explanation of our strategy is shown as follows.

The projection of space $\Omega$ from a specific viewpoint allows us to obtain a normalized z-buffer image in which each pixel contains the number of occupied points $q$ which are projected into it. We then define a normalized projected image. The pixel value goes from 0, which means that there is no occupied-projected point at this location, to 1 for the highest number of occupied-projected points. We will denote these images as $I_\Omega(\alpha, \beta)$, where $\alpha$ and $\beta$ are azimuth and elevation angles of the projection. In indoor environments, as in our case, the projection directions should correspond to vertical and horizontal views. Figure 4 a) illustrates images $I_\Omega$ from top-down and horizontal viewpoints. Note that the walls of the room are clearly distinguished in a bright color. A color version is shown in part b), in which we can see the number of projected points per pixel.

A thresholding process can be used to extract the segments that represent the walls in this image and to obtain the corresponding points in spaces $\Omega$ and $\Gamma$. The planes corresponding to the walls in the $\Omega$ coordinate space are called Zones 0 ($Z_0$), whereas the planes in front of and behind it are denoted as $Z_{-1}, Z_{-2}, \ldots$ and $Z_1, Z_2, \ldots$. Of course, a plane in space $\Omega$ corresponds to a parallelepiped in space $\Gamma$. It is therefore eventually possible to deal with the 3D data belonging to different zones, especially those in Zone 0. Figure 4 below shows Zones 0 for the walls of a room. Bear in mind that this zonal representation allows us to discover the 3D data in the neighborhood of the walls for different voxel sizes. In spite of this, it is possible to make an analysis of the nearby objects that might occlude the wall. For instance, Figure 5 shows the 3D data in Zone 0 on the left and in zones $Z_{-2}, Z_{-1}, Z_0, Z_1, Z_2$ (the colors magenta, yellow, red, blue and green) for 5 cm and 10 cm voxel sizes on the right. Note several zones at the top of the wall which remain white. Those points might correspond to zones which are not reached by the laser scanner from any position and no data are therefore available in these locations.

Figure 4. Wall detection. Above) Voting projected images $I_\Omega(90,0)$ and $I_\Omega(0,0)$ in grey and color versions. Below) Walls obtained in the voxel representation space.
4.2 Voxel labeling on the wall

As was mentioned in Section 2, voxel representation is frequently used in NBV and view planning. A voxel representation creates a fixed topology in which certain criteria for the subsequent positions of the laser scanner are established [Blaer 06], [Blaer 07], [Blaer 06-2] [Mass-98]. In the majority of approaches, the occlusion problem is one of the key factors to be taken into account. Other authors use an octree or mesh representation to model the occlusion [Low 2006] [Low 2006-2]. In our case, 3D data is provided in advance from different sensor locations and we do not therefore tackle the occlusion problem as a typical NBV issue. Nevertheless, since we also aim to label the scene – or more specifically, the walls in the scene
– by using 3D range data coming from different viewpoints, we have implemented our approach in a 3D voxel representation framework.

Once the walls have been delimited, we carry out the labeling of the voxels which are inside the visual space from each sensor position. Formally, let \( o_j \) be the location of the \( j \)-th sensor in \( \Omega \) and \( q_1, q_2, q_3 \) and \( q_n \) the vertex of a certain extracted wall. The space to be labeled then corresponds to the volume \( S_j = \{o_j, q_1, q_2, ..., q_n\} \) where the basis of \( S_j \) corresponds to the convex polygon \( Z_0 = \{q_1, q_2, ..., q_n\} \) (Figure 6 a).

Volume \( S_j \) is labeled through the use of a ray tracing approach from \( o_j \). A generic ray \( \Pi \) which goes from \( o_j \) to any voxel of \( Z_0 \) is used to classify all the crossing voxels into three types: empty voxel (A), occupied voxel (B) and occluded voxel (C). The occupied voxels have been previously labeled when the space \( \Omega \) is built. All voxels that are between \( o_j \) and an occupied voxel, if any, are labeled as empty voxels. In the case of a non-occupied voxel in the ray, all voxels will be empty. Finally, all voxels between an occupied voxel and \( Z_0 \) are labeled as occluded voxels. Although we are only interested in the labeling of \( Z_0 \) in this work, all labeling information might be used in future extensions of the method.

The implementation of the voxel labeling is carried out in a discrete space. In order to decide whether or not the ray collides with a voxel, we consider voxels as cubes with length \( l \). Thus, a ray \( \Pi \) goes across voxels which are closer than \( l/\sqrt{2} \). Obviously this discretization of the space may cause errors in the labeling of \( Z_0 \). Note that an intermediate occupied voxel might affect several rays, especially if the voxel is near the sensor position. Consequently, over-occlusion effects appear and occupied voxels in \( Z_0 \) might become occluded. However, these errors are resolved after integrating all views and re-labeling the occupied voxels in \( Z_0 \), which are set at the beginning.

Figure 6 a) illustrates details of voxel labeling. A specific ray is plotted in space \( \Omega \) by assigning different colors to each label type. Figure 6 b) and c) show the voxel labeling results for a set of volumes \( S_j, j=1,...,5 \) for a specific wall of the room. In this case, five range data corresponding to five laser scanner positions are available. The colors green, red and blue correspond to empty, occupied and occluded voxels respectively. In this figure, a part of the reflectance images including the selected wall is also inserted.
After labeling a set of volumes $S_j$, $j=1,...,k$, corresponding to a certain wall, we have $k$ labels for each voxel of $Z_0$. In the next stage we therefore integrate all labels, thus yielding a definitive label for each voxel of the wall. Let $\{l_1, l_2, ..., l_k\}$ be the labels of a voxel $v$ of $Z_0$ from laser scanner positions $\{o_1, o_2, ..., o_k\}$, where $l_j \in \{\text{empty, occupied, occluded}\}$, $j = 1,2,...,k$, and let $l_0$ be the label of $v$ before the raytracing process, $l_0$ being $\{\text{empty, occupied}\}$. The label of voxel $v$ is calculated by following a priority algorithm in three steps:

- I) $l(v) = \{\text{empty}\}$, if $l_0 = \{\text{empty}\}$ and $\exists j$, $j = 1,2,...,k$  $l_j = \{\text{empty}\}$
- II) $l(v) = \{\text{occluded}\}$, if $l_0 = \{\text{empty}\}$ and $l_j = \{\text{occluded}\}$, $\forall j = 1,2,...,k$
- III) $l(v) = \{\text{occupied}\}$, if $l_0 = \{\text{occupied}\}$ or if $\exists j$, $j = 1,2,...,k$, $l_j = \{\text{occupied}\}$

Figure 7 shows labeling from each scanner position and the results after integration. Note that initial occupied voxels maintain the label after the label fusion algorithm. In Figure 7 right, the labeling of $Z_0$ is presented as overlapping spaces $\Omega$ and $\Gamma$. In this picture, 3D data is plotted in black whereas the 3D points corresponding to the centers of the voxels are painted in the label colors. Owing to discretization of the space several disagreements can be seen. For example, red points, which correspond to occupied voxels, are in the middle of sparse 3D regions. The reason for this is that a voxel is occupied if it contains a single 3D point. This circumstance is illustrated in the figure.
Figure 7. Discrete representation of the voxel labeling from five laser scanner positions, labeling integration and labeling mapping in 3D data space $\Gamma$.

4.3 3D data labelling in the wall

After classifying the voxels of $Z_0$ in space $\Omega$, the following step consists of translating this information to space $\Gamma$. In order to be able to apply 2D image processing in the rest of our approach, it is first necessary to represent voxels and 3D data information of the wall in a 2D image context. We therefore represent the voxel labeling on the wall as a 2D image $\Omega$ where, assuming that $\mu_\Omega$ is the voxel size, each pixel represents $\mu_\Omega \times \mu_\Omega$ centimeters in space $\Gamma$ and contains one assigned label.

On the other hand, in order to label space $\Gamma$ as accurately as possible, 3D data is mapped onto the highest resolution 2D master image, which we will denote as $I$. From the outset, $I$ is therefore a high resolution image which will contain only occupied labels corresponding to the 3D points in Zone 0. The voxel labels in $I_\Omega$ are then also integrated into $I$. Note that the resolution of $I$, $\mu_\Omega$ is much higher than $\mu_\Omega$ so that our goal here is to label the entire image $I$ by taking the voxel labels as seeds in a further growing algorithm. Figure 8 illustrates an example. In part a) red, blue and green pixels correspond to the labels of the voxels which have been mapped onto $I$, black pixels which are those that are occupied by the 3D data, and white pixels are those that must be labeled though the growing algorithm. Consequently, at this point, we add the new label ‘unknown’ for such a set of white points. Note that an unknown pixel might become occluded or empty, but never occupied. In other words, if an unknown pixel is not converted to occluded or empty, we can therefore conclude that it is a non-sensed pixel. Non-sensed pixels should consequently be added to the occluded ones later in the hole-filling stage.

Let us assume the original image $I$ with labels A (empty), B (occupied), C (occluded) and D (unknown). We accomplish a sequential region growing algorithm by taking occluded and
empty pixels as seeds and introducing various growing conditions. The growth of occupied pixels is not appropriate because it would lead to a modification of the set of original data. The growth of a generic occluded region $O$ is accomplished as follows:

- $p_0(x,y) \in O$ if $I(x,y)=C$
- $p_{t+1}(x,y) \in O$ if $I_{t+1}(x,y)=C$ or $I_{t+1}(x,y)=D$ verifying
  \[
  \min\{d(I_{t+1}(x,y), I^A)\} > \alpha \& \min\{d(I_{t+1}(x,y), I^C)\} < \beta
  \]

For empty regions the algorithm is as follows:

- $p_0(x,y) \in O$ if $I(x,y)=A$
- $p_{t+1}(x,y) \in O$ if $I_{t+1}(x,y)=A$ or $I_{t+1}(x,y)=D$ verifying
  \[
  \min\{d(I_{t+1}(x,y), I^C)\} > \alpha \& \min\{d(I_{t+1}(x,y), I^A)\} < \beta
  \]

$I^A$ and $I^C$ are the sets of pixels labeled as $A$ and $C$ in image $I$, and $d(p,S)$ is the Euclidean distance operation between $p$ and any element of the set $S$. Thresholds $\alpha$ and $\beta$ are empirically established in these expressions.

As a result of this labeling growing process, image $I$ updates the labels assigned to all pixels, which have new $A$ (empty), $B$ (occupied), $C$ (occluded) and $D$ (non-sensed) labels (see Figure 8 b)). As was previously mentioned, label $D$ corresponds to non-sensed locations in space $\Gamma$ or to occluded regions which have not been correctly identified for reasons of space discretization (See Figure 8 b)). Consequently, in terms of occlusion, D labels can be added to C labels (Figures 8 c)).

![Figure 8. 3D data labeling. a) Voxel label mapping over image $I$. b) Result of the label growing algorithm. c) Grouping of occluded and non-sensed pixels.](image)

5. Segmentation of Occluded Regions

5.1 Statement of the problem

Since image $I$ provides the necessary information with regard to occlusion, a hole-filling algorithm could be applied in order to reconstruct the wall. Nevertheless, before applying a hole-filling algorithm, it is necessary to know which part of each occluded region belongs to the wall and which does not. Then, we will deal with the refilling of the occluded zones belonging to the wall. It is therefore necessary to again re-label an occluded region (type C) as *Occluded*
region belonging to the wall (C1) and Occluded region not belonging to the wall (C2). Figure 9 shows an example in which the occluded region covers part of a bookshelf, part of the wall and part of the door. Note that C2 region corresponds to the gaps in the wall (in this case, to a window) and that these are not, therefore, a part of the wall structure. It is for this reason that they will from now on be denominated as holes. Basically, the most important holes in a wall are doors, windows, closets and built-in shelves. In this work, we tackle the problem of recognizing this type of object. Of course, future works might address extending the recognition problem to other objects that may frequently appear on walls, such as pictures, whiteboards, friezes, a bookshelf in front of the wall, and so on.

Several papers regarding door and window detection in a less general context can be found in journals and conferences. Most of them operate on 2D images and are adapted to mobile robot applications (SLAM, navigation and positioning) [Lee 04] [Muño 06] [Chen 08]. Furthermore, the majority of them do not consider occlusion circumstances, which highly reduce the recognition problem complexity. One of the most recent papers regarding this subject corresponds to the reference [Ali 08]. Ali et al. use range information and develop a statistical method that exploits basic local features, such as mean, variance, and standard deviation of the distance data. They binarize the image using an adaptive thresholding algorithm and extract the resulting bounding rectangles, which are used to retrieve the positions of windows in the image, but they do not consider occlusion.

![Figure 9. Detail of an occluded region in the wall and label segmentation in labels C1 (Occluded region belonging to the wall) and C2 (Occluded region not belonging to the wall).](image)

In the following paragraphs we present a hole taxonomy for walls in occlusion terms and for the environment in which we are working. Figure 10 illustrates such a taxonomy. Let \( \Theta \) be a hole in a wall. We classify this by taking into account the type of occlusion and the corresponding labels in \( \Theta \) as follows:

Case 1. \( \Theta \) is not occluded
1.1 If \( \Theta \) is open \( \Rightarrow \) \( \Theta \) will be identified in regions A (empty region)
1.2 If \( \Theta \) is totally closed \( \Rightarrow \) \( \Theta \) will be identified in regions B (occupied region)
1.3 If \( \Theta \) is partially closed (for example, as a result of curtains, blinds, shutters) \( \Rightarrow \) \( \Theta \) will be identified in regions A and B.

Case 2. \( \Theta \) is occluded
2.1 If \( \Theta \) is open \( \Rightarrow \) \( \Theta \) could be identified in regions A (empty) and C (occluded)

17
2.2 If Θ is totally closed ⇒ Θ could be identified in regions B (occupied) and C (occluded)
2.3 If Θ is partially closed (for example, as a result of curtains, blinds, shutters) ⇒ Θ will be identified in regions A, B and/or C.

Figure 10. Occlusion taxonomy in our labeling environment

5.2 Feature extraction on the wall

Assuming that doors and windows are nearly rectangular shapes in image \( I \), which is a reasonable hypothesis, the first stage in our hole recognition approach consists of extracting a set of candidate rectangles on the wall. To do this, we compute the [0,1] normalized range image \( I_p \) of the Zone 0, which has the same dimension as \( I \), and carry out an edge detection processing on it. A thresholding technique over horizontal and vertical edge profiles in the image allows us to identify the essential horizontal and vertical lines in the image. Finally, all combinations of crossing points are used to establish a wide set of hole candidates. Figure 11 shows several steps with regard to this process.
Our hole identification approach is based on the use of supervised learning algorithms that consider a set of features for each rectangle candidate. The features not only relate to absolute and relative geometrical characteristics on the wall but also include labeling information contained in image $I$. Thus, we define a fourteen component feature vector $V = \{v_1, v_2, ..., v_{14}\}$ as follows:

$v_1$ – Absolute area  
v_2$ – Width/Height ratio  
v_3, v_4$ – Relative sizes on the wall: $\text{dimension1}(\Theta)/\text{dimension1}(Z_0), \text{dim2}(\Theta)/\text{dim2}(Z_0)$  
v_5, v_6, v_7, v_8$ – Distances from the sides to the edges of the wall  
v_9$ – Flatness: root mean squared error to the best plane fitted to the rectangle surface  
v_{10}, v_{11}, v_{12}$ – A, B and C labeling percentages inside the rectangle.  
v_{13}$ – Number of interior rectangles  
v_{14}$ – Number of interior inverted-U rectangles

Figure 12 illustrates each of the features of vector $V$. Feature $v_{13}$ takes into account segments in the image that could belong to a window frame. In the ideal case, a window frame is defined by a pair of vertical segments joined to a pair of horizontal segments. Nevertheless, on other occasions, and owing to the fact that various deep discontinuities in the frame structure arise, it is possible to detect two or more concentric frame edges. We include this characteristic through feature $v_{13}$. The same case might occur in doors which have several inverted-U rectangles inside an external rectangle. Feature $v_{14}$ concerns these cases (See Figure 12 bottom right).
5.2 Hole Recognition

The hole candidates from image $I_p$, have been classified through the use of a Support Vector Machine [Vapnick 1998] classification algorithm in which the feature vector $V = \{v_1, v_2, ..., v_{14}\}$ is used as an input. It is well known that Support Vector Machine (SVM) is a learning system that performs classification tasks by constructing hyper planes in a multidimensional space. We have used a Radial Basis Function (RBF) as a kernel in a binary SVM model.

Since a supervised learning algorithm is being dealt with, a set of known labels is needed for training. It is thus necessary to obtain ground truth labels for a set of training examples. To date we have taken two different strategies. Firstly, as in many referenced works regarding occlusion in images [Ali 2008], [Hoiem 2008], we carried out a manual segmentation to build the ground truth. The learning set is obtained by manually identifying the holes’ corners in images $I$ and thus defining a set of initial hole models. One disadvantage of this strategy is that it is necessary to label the holes in training data and that this would need to be done for each type of being modeled. A second idea is to create a method that does not require labeled training data. In this case, we automatically identify some easy cases in which there is no occlusion, and create models of a particular type of holes. These models could then be used to recognize difficult cases with occlusion. Such cases are easily identified and correspond to, for example, open doors and unoccluded windows in the wall.

Only the first procedure has been developed to date. In order to consider the uncertainty of the detected positions in both cases, we add new models by injecting Gaussian noise into the initial
set of hole models. We are thus able to achieve learning databases of hundreds of holes in the ground truth. We assigned 80% of the learning database to train the system and 20% to test whether the learning is reliable.

The holes in a query room are recognized by following all the 3D data processing stages explained in the previous sections: voxel representation, wall identification in the room, wall voxel labeling, 3D data labelling on the walls, candidate rectangle extraction and feature extraction. Once a query image $I_p'$ with $m$ rectangles and their corresponding feature vectors $V'_1, V'_2, ..., V'_m$, have been obtained, the candidates are then classified as holes or otherwise according to the decision function obtained after running the SVM learning algorithm. The output of the SVM method thus classifies a rectangle of $I_p'$ as a hole or otherwise.

Since many rectangles could be initially extracted in image $I$, several close holes might appear for the same real gap. This frequently occurs near the door and window boundaries owing to the fact that their frames cause depth discontinuities in the image $I_p$. New lines and new rectangles consequently appear. However, this is not a drawback because we are working with complex scenes in which occlusion is an important source of uncertainty, and no candidate can be discarded since it might eventually be a true solution.

Therefore, rather than rejecting candidates, we carry out a clustering process over all holes yielded by the SVM algorithm and take the prototype of each cluster as the definitive hole. This grouping is accomplished through the use of a hierarchical clustering algorithm, taking the coordinates of the vertices of the rectangles in the wall reference system as input data. Thus, we find a partition in which holes within each cluster are as close to each other as possible, and as far from holes in other clusters as possible. The result is a set of clusters that are compact and well-separated. The optimum number of clusters is calculated through an iterative procedure that finds the global minima of the sum of distances within clusters for each hypothesis concerning the number of clusters. For example, assuming that there are no more than $h$ holes in a room of the building, we check the results of the clustering algorithm from 2 to $h$ clusters and choose that which yields the lowest sum of distances within clusters. Figure 13 shows an example of the hole recognition approach on a wall. In part a) the objects that SVM classifies as holes are plotted over image $I$ (colored in gray tones). Part b) shows the hole groups, each in a different color, after applying the clustering algorithm. Part c) illustrates the set of final holes in the wall.

The hole extraction stage yields enough information to accomplish the segmentation of occluded parts, as was mentioned in Section 5. Thus occluded parts, originally labeled in image $I$ as C, are now relabeled as $C_1$ (hole) if the pixels belong to any hole, and $C_2$ if not. In practice, pixels $C_2$ keep the label name ‘occluded’ and pixels $C_1$ are finally labeled as hole pixels. Figure 13 d) shows the hole borders superimposed onto $I$. Note that the glass in the windows is behind the wall. Thus, the points corresponding to some of the papers that appear attached to the glass have fallen out of Zone 0 and are consequently labeled as empty rather than occupied. Figure 13 e) presents the final labeling of the wall.
6. Filling Occluded Parts of the Wall

The gap filling technique that we use here was published in [Perez 2008]. We propose the use of image inpainting algorithms, traditionally used to recover and restore damaged photos. In our case the algorithm is applied on the depth image \( I_p \) where the gaps correspond to occluded zones. The goals and applications of inpainting extend from the restoration of damaged paintings and photographs to the removal/replacement of selected objects. We apply a 3D version of the original image inpainting algorithm proposed by Stefan Roth and Michael J. Black [Roth 2005]. Most of these kinds of algorithms usually obtain the inpainted data by using the original damaged image.

The technique published in [Perez 2008] was originally applied to the filling of gaps in 3D meshes. Since these algorithms need an image as input, the first stage of the method concerned a 3D to 2D transformation. By projecting the 3D surface onto a squared plane, a 2D image was generated in such a way that the depth information was stored in each grid. For the case presented in this paper, this stage has already taken place, since the 3D to 2D transformation has been carried out and the depth information is contained in image \( I_p \). Furthermore, the regions to be filled are identified in image \( I \). The image restoration algorithm is then applied on \( I_p \). Once the image has been repaired, the inverse transformation 2D to 3D is performed and the repaired wall is obtained.

The Roth et al. algorithm is based on Markov Random Fields, signifying that it basically learns the data to be inpainted from the response of a bank of images to a set a filters. In our case the algorithm has been trained on 2000 randomly cropped image regions that are taken from fifty images from the Berkeley Segmentation Database (natural scenes, people, buildings, etc.) [Mart 2008].
Before applying the filling algorithm, several filters are applied in order to remove the noise of the original data, and the spikes which appears on the gap boundaries. Figure 14 a) and b) show \( f_p \) and the preprocessing for an example.

![Figure 14. a) Depth image and 3D representation of Zone 0. b) Filtering process to remove noise and spikes c) Result after applying the filling algorithm.](image)

After the point filling is completed, several refinement stages are carried out. On the one hand, since the reconstructed surfaces appear to be unnaturally smooth we have added artificial Gaussian noise for aesthetic reasons. This was done by calculating the standard deviation in the depth axis for the wall surface and injecting the same amount of Gaussian noise into the filled regions. On the other hand, a mesh and rendered representation of the wall is accomplished. Figure 14 c) presents the final result.

7. Experimental Results

This section shows the experimental results obtained after testing our approach on panoramic range data belonging to the interior of a three floor building. Our method was applied to different sets of range images taken in different rooms of a building, which was scanned by a professional service provider, Quantapoint, Inc. We used this test-set to evaluate the proposed 3D indoor reconstruction method. Figure 15 shows various panoramic images corresponding to different rooms in the building. Note that we are dealing with very complex scenarios with a high degree of clutter and occlusion. They contain a wide variety of objects that occlude not only the walls but also the ceiling and floor. Three to six laser scans were taken per room.
We have developed our method in Matlab 7.2 on a Pentium III computer. It was necessary to work with a reduced data set owing to the evident computational constraints of such a development environment. The original 3D data were thus resampled by a factor of 1:25. Despite this, we worked with a cloud of over two millions points per room. Owing to reasons of poor surface reflectance and the fact that no NBV method had been used in the data acquisition phase it is possible to find non-sensed small regions on the walls. The corresponding points are initially labeled as type D (*unknown*) and are finally refilled by following the proposed hole-filling algorithm. This is clearly shown in Figure 16. The image corresponds to Zones -2,-1,0,+1,+2 of the floor and ceiling in which many points are missing.

To date, we have tested the method for voxels of 5 and 10 cm dimensions in spaces of around 400 m³, which involves managing 342,400 and 2,709,360 voxels respectively. This amount of data is within the bounds that our algorithm is currently able to manage. With regard to the
wall’s data, the average wall dimensions are 25 to 78 m² and the resolution corresponding to images $I$ is 2 cm per pixel. The walls are therefore eventually reconstructed in 2 cm grids. Table 1 summarizes some interesting information regarding the voxel and wall labeling phases for a room with five sensor positions (Figure 15). We include Number of 3D points on the wall ($N3D$), size of the wall in m² ($SW$), accuracy in cm of the image $I$ ($AI$), dimension of image $I$ ($DIM$) and the total number of rays in the raytracing stage ($NR$).

The process is relatively slow, owing to the fact that it run without optimizing the computational cost. It takes about three and fifteen minutes per wall for voxelizations of 10 and 5 cm respectively. The largest contribution to total processing time comes from the raytracing stage. Both notable time improvements and higher accuracy (reducing the size of voxel space) are expected after recoding the algorithm in a compiling machine.

Table 1. Voxel labeling stage data.

<table>
<thead>
<tr>
<th>Scene</th>
<th>$N3D$</th>
<th>$SW$</th>
<th>$AI$</th>
<th>$DIM$</th>
<th>$NR$</th>
<th>$N3D$</th>
<th>$SW$</th>
<th>$AI$</th>
<th>$DIM$</th>
<th>$NR$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene1</td>
<td>352485</td>
<td>37.7508</td>
<td>0.02</td>
<td>489x193</td>
<td>19305</td>
<td>191556</td>
<td>37.5552</td>
<td>0.02</td>
<td>489x192</td>
<td>75460</td>
</tr>
<tr>
<td>Scene2</td>
<td>165317</td>
<td>37.6736</td>
<td>0.02</td>
<td>488x193</td>
<td>19800</td>
<td>140900</td>
<td>37.6736</td>
<td>0.02</td>
<td>488x193</td>
<td>76440</td>
</tr>
<tr>
<td>Scene3</td>
<td>152529</td>
<td>25.862</td>
<td>0.02</td>
<td>335x193</td>
<td>13600</td>
<td>152467</td>
<td>25.862</td>
<td>0.02</td>
<td>335x193</td>
<td>52650</td>
</tr>
<tr>
<td>Scene4</td>
<td>194265</td>
<td>25.9392</td>
<td>0.02</td>
<td>336x193</td>
<td>13600</td>
<td>190525</td>
<td>25.9392</td>
<td>0.02</td>
<td>336x193</td>
<td>52650</td>
</tr>
<tr>
<td>Scene5</td>
<td>253832</td>
<td>66.1128</td>
<td>0.02</td>
<td>489x338</td>
<td>34155</td>
<td>24926</td>
<td>65.6476</td>
<td>0.02</td>
<td>487x337</td>
<td>133280</td>
</tr>
<tr>
<td>Scene6</td>
<td>182694</td>
<td>78.0192</td>
<td>0.02</td>
<td>516x378</td>
<td>39520</td>
<td>24926</td>
<td>65.6476</td>
<td>0.02</td>
<td>487x337</td>
<td>157320</td>
</tr>
</tbody>
</table>

Figure 17 a) and e) shows examples of original 3D data and the reconstruction of the walls, taking Zone 0’s depths of 5 and 10 centimeters respectively. As can be seen, the parts of the wall hidden behind furniture and facility pieces are totally reconstructed. Parts b) c) and d) show details of the results of the different stages in the process: labeling for the hole-filling process and the result of the hole-filling algorithm. Note that the size of the voxels in space $\Omega$ might have an influence on the number of holes identified, which is more accurate in the case of 5 cm. The size of the voxel in such a representation is a crucial point in the final result and should be carefully evaluated, since taking the smallest size might not always be the best solution. Bear in mind that, for small sizes, we run the risk that Zone 0 will contain only a part of the wall. For example, in the case of an insufficient wall flatness level or if a slight error is made in the wall detection step, many data will fall out of Zone 0 and these points will therefore be missed in the rest of the process. On the other hand, large voxel sizes give an imprecise solution because some occlusions might not be identified. For example, if a 10 cm size is taken in Scene 1, occlusions of door frames and skirting boards are not detected.
Figure 17. Final results for Scenes 1, 2 and 4. a) Rendered 3D model of initial Zone 0 b) Labeling for the hole-filling process c) Hole-filling algorithm result. Red points correspond to the added points. d) Rendered model after filling the occluded parts. e) Final rendered model after extracting the holes.

All reconstructions in Figures 17 appear to be visually acceptable. In order to quantify the performance of our approach, we have generated statistics based on the number of true and false occluded region that the algorithm detects, taking two different voxel sizes in the process. To do this, true occlusions have been evaluated by using the reflectance maps corresponding to different positions of the sensor. We also present the relative percentages of labels A, B, C and D before the hole detection phase. All these results are presented in Table 2. We were unable to count the number of true occluded segments in Scenes 5 and 6, which correspond to ceiling and floor surfaces.
Table 2. Occlusion data and 3D labeling before the hole recognition phase. Occlusion data and 3D labeling: NT (Number of true occluded segments), NTD (Number of true detected occluded segments), NFD (Number of false detected occluded segments).

<table>
<thead>
<tr>
<th>Scene</th>
<th>Vox 10cm NT</th>
<th>NTD</th>
<th>NFD</th>
<th>A%</th>
<th>B%</th>
<th>C%</th>
<th>D%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene1</td>
<td>17</td>
<td>12</td>
<td>2</td>
<td>24.6</td>
<td>60.6</td>
<td>9.9</td>
<td>4.7</td>
</tr>
<tr>
<td>Scene2</td>
<td>16</td>
<td>14</td>
<td>1</td>
<td>24.4</td>
<td>48.1</td>
<td>26.9</td>
<td>0.5</td>
</tr>
<tr>
<td>Scene3</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>0.01</td>
<td>51.6</td>
<td>38.6</td>
<td>9.6</td>
</tr>
<tr>
<td>Scene4</td>
<td>10</td>
<td>9</td>
<td>2</td>
<td>21.5</td>
<td>52.8</td>
<td>18.4</td>
<td>7.2</td>
</tr>
<tr>
<td>Scene5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.7</td>
<td>68.9</td>
<td>2.6</td>
<td>24.7</td>
</tr>
<tr>
<td>Scene6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.44</td>
<td>46.1</td>
<td>29.2</td>
<td>20.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scene</th>
<th>Vox 5cm NT</th>
<th>NTD</th>
<th>NFD</th>
<th>A%</th>
<th>B%</th>
<th>C%</th>
<th>D%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene1</td>
<td>17</td>
<td>18</td>
<td>1</td>
<td>26.63</td>
<td>40.41</td>
<td>29.02</td>
<td>3.95</td>
</tr>
<tr>
<td>Scene2</td>
<td>16</td>
<td>16</td>
<td>1</td>
<td>31.17</td>
<td>39.61</td>
<td>26.22</td>
<td>3.01</td>
</tr>
<tr>
<td>Scene3</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>0.43</td>
<td>51.53</td>
<td>43.72</td>
<td>4.32</td>
</tr>
<tr>
<td>Scene4</td>
<td>10</td>
<td>9</td>
<td>2</td>
<td>24.56</td>
<td>52.20</td>
<td>19.69</td>
<td>3.55</td>
</tr>
<tr>
<td>Scene5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12.12</td>
<td>68.88</td>
<td>2.84</td>
<td>16.37</td>
</tr>
<tr>
<td>Scene6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11.44</td>
<td>44.16</td>
<td>30.67</td>
<td>13.74</td>
</tr>
</tbody>
</table>

With regard to the hole detection stage, we have made an evaluation by counting a **true positive** if the existing hole is detected, a **false positive** if a nonexistent hole is detected, and a **true negative** if the existing hole is not detected. We also included the relative percentages of...
labels corresponding to the filling phase. As was mentioned in Section 5.3, as far as the hole-filling phase is concerned, only three label types (hole (A), occupied (B) and occluded (C)) are finally set in image $I$, this being the space corresponding to that filled with pixels C.

At this point it is important to highlight several aspects that demonstrate the efficiency of our method in difficult cases. Note that in Scene 2 (Figure 17), the hole is correctly recognized in spite of the existence of occupied and occluded regions inside several windows. Occupied zones corresponding to blinds and air-conditioning match, whereas occluded areas are caused by objects in front of the wall. Furthermore, the algorithm also worked correctly in negative cases. Thus, although Scenes 5 and 6 (corresponding to the ceiling and floor) do not have holes, they have empty regions (type A), which could be thought of as potential holes. However, no hole was recognized in this case. The appearance of these regions is caused by the lack of flatness on the ceilings and floors. Thus, Zone 0 is more imprecise and this causes several points of the wall to fall out of the Zone 0 border.

Table 3. Hole detection and final labeling. Hole detection and final labeling: NG (Number of holes), NTP (Number of true positive), NFP (Number of false positive), NTN (Number of true negative), A (hole), B (occupied), C (to be filled).

<table>
<thead>
<tr>
<th>Vox 10cm</th>
<th>NG</th>
<th>NTP</th>
<th>NFP</th>
<th>NTN</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene1</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>21.0</td>
<td>57.0</td>
<td>21.0</td>
</tr>
<tr>
<td>Scene2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>39.0</td>
<td>36.0</td>
<td>23.0</td>
</tr>
<tr>
<td>Scene3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50.7</td>
<td>49.3</td>
</tr>
<tr>
<td>Scene4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>30.5</td>
<td>49.5</td>
<td>19.9</td>
</tr>
<tr>
<td>Scene5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68.9</td>
<td>31.1</td>
</tr>
<tr>
<td>Scene6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>46.1</td>
<td>53.9</td>
</tr>
<tr>
<td>Vox 5cm</td>
<td>NG</td>
<td>NTP</td>
<td>NFP</td>
<td>NTN</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Scene1</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>13.5</td>
<td>35.9</td>
<td>32.5</td>
</tr>
<tr>
<td>Scene2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>41.5</td>
<td>34.6</td>
<td>24.3</td>
</tr>
<tr>
<td>Scene3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50.5</td>
<td>49.5</td>
</tr>
<tr>
<td>Scene4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>30.6</td>
<td>49.4</td>
<td>19.8</td>
</tr>
<tr>
<td>Scene5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68.8</td>
<td>31.1</td>
</tr>
<tr>
<td>Scene6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>44.1</td>
<td>55.8</td>
</tr>
</tbody>
</table>

8. Conclusions

The identification and restoration of parts of walls hidden behind furniture and cluttered objects in complex interiors is a challenging question that must be studied in depth. In this paper we demonstrate that the completion of this information involves a careful and difficult process in which several keys must be efficiently resolved. To date, partial solutions in the reconstruction of surfaces under occlusion in range images have been published in literature, most of them in simple non-panoramic scenarios.

It could be assumed that the solution to the occlusion problem in interiors would consist of taking additional scans of the scene. Nevertheless, in many publications concerning scan planning it is demonstrated that, even when tens of range data from different viewpoints exist, the scene is incomplete. It is thus worth developing algorithms which reconstruct parts of the missing regions rather than attempting to observe them.

The method presented in this paper has been tested on a real interior scenes in which up to six range images per room are available. The results lead us to state that most of the occluded surfaces on the wall were detected and filled. The detection of holes was successful in the majority of cases, depending on the voxel size. Thus, an acceptable final wall reconstruction, through a 3D hole-filling approach, is carried out at the end of the process. In summary we can
conclude that future promising results are expected in 3D interior reconstruction by following the guidelines of this technique.

Future improvements to the method will be addressed in various lines. Firstly, in spite of the high complexity of the scenes tested in this work, in order to improve the efficiency of the algorithms and make them more robust, a more extended test should be carried out in other scenarios over the next few months. Secondly, moving the codes to a compiled programming language will yield several benefits. Obviously this will improve the processing time in all the stages, especially in the raytracing phase. Also, a higher amount of 3D data could be processed and smaller voxel sizes could be used. With regard to the hole recognition, the learning set should be better implemented by incorporating new ideas with the aim of creating automatic learning models. Another interesting idea concerns the use of reflectance images to make the method more robust. For example, in the first steps of the hole identification phase, lines and rectangles are extracted from the depth image of the wall $I_w$. This is a critical point in which reflectance information could help to make this extraction more accurate and reliable.

References


[SAPPA 2002] A. D. Sappa. Improving Segmentation Results by Studying Surface Continuity. 16th International Conference on Pattern Recognition (ICPR'02), Volume 2, Quebec City, QC, Canada, August 2002.


