Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data

Xuehan Xiong[b], Antonio Adan[a], Burcu Akinci[c], Daniel Huber[b,*]

[a] *Department of Electrical Engineering, Electronics, and Automation, Castilla La Mancha University, Ciudad Real, Spain*
[b] *The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA*
[c] *Department of Civil and Environmental Engineering, Carnegie Mellon University, USA*

Abstract

In the Architecture, Engineering, and Construction (AEC) domain, semantically rich 3D information models are increasingly used throughout a facility's life cycle for diverse applications, such as planning renovations, space usage planning, and managing building maintenance. These models, which are known as building information models (BIMs), are often constructed using dense, three dimensional (3D) point measurements obtained from laser scanners. Laser scanners can rapidly capture the "as-is" conditions of a facility, which may differ significantly from the design drawings. Currently, the conversion from laser scan data to BIM is primarily a manual operation, and it is labor-intensive and can be error-prone. This paper presents a method to automatically convert the raw 3D point data from a laser scanner positioned at multiple locations throughout a facility into a compact, semantically rich information model. Our algorithm is capable of identifying and modeling the

*Corresponding author
Email addresses: xxiong@andrew.cmu.edu (Xuehan Xiong), Antonio.Adan@uclm.es (Antonio Adan), bakinci@cmu.edu (Burcu Akinci), dhuber@cs.cmu.edu (Daniel Huber)

main visible structural components of an indoor environment (walls, floors, ceilings, windows, and doorways) despite the presence of significant clutter and occlusion, which occur frequently in natural indoor environments. Our method begins by extracting planar patches from a voxelized version of the input point cloud. The algorithm learns the unique features of different types of surfaces and the contextual relationships between them and uses this knowledge to automatically label patches as walls, ceilings, or floors. Then, we perform a detailed analysis of the recognized surfaces to locate openings, such as windows and doorways. This process uses visibility reasoning to fuse measurements from different scan locations and to identify occluded regions and holes in the surface. Next, we use a learning algorithm to intelligently estimate the shape of window and doorway openings even when partially occluded. Finally, occluded surface regions are filled in using a 3D inpainting algorithm. We evaluated the method on a large, highly cluttered data set of a building with forty separate rooms.

Keywords:
interior modeling, 3D modeling, scan to BIM, lidar, object recognition, wall analysis, opening detection.

---

1. Introduction

In the Architecture, Engineering, and Construction (AEC) domain, semantically rich 3D models are increasingly used throughout a building's life cycle, from design, through construction, and into the facility management phase. These models, generally known as building information models (BIMs), are used for many purposes, including planning and visualization

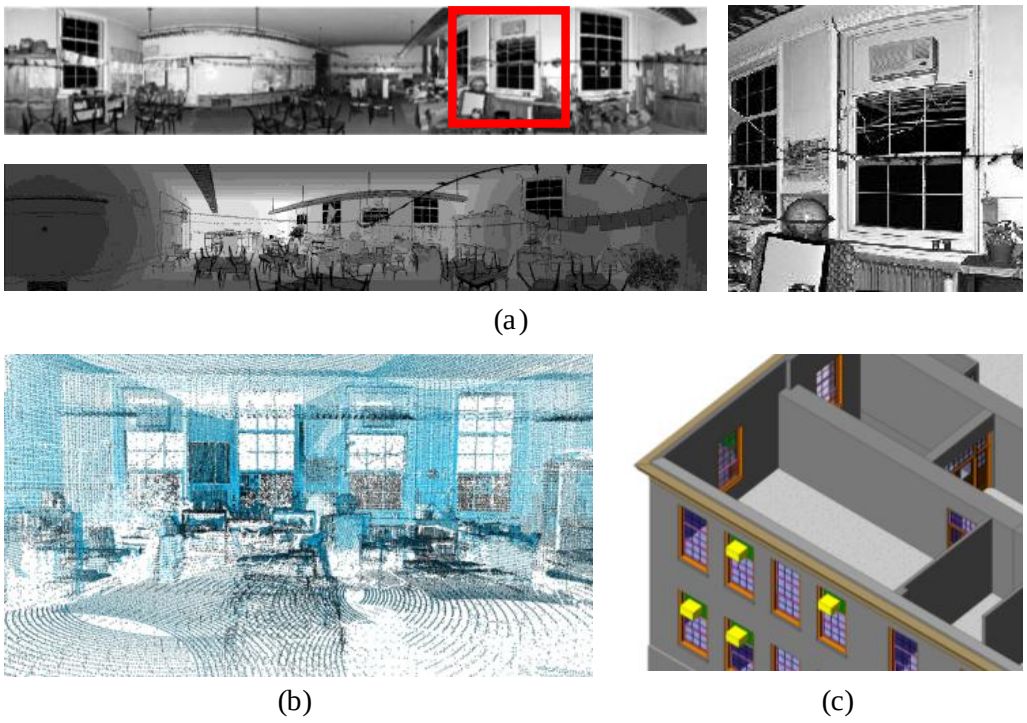


Figure 1: As-is BIM creation. a) The input data from a single scanner location reflectance image (top), range image (bottom), and close up of highlighted region (right). b) The point data is aligned and combined into a cloud of points. Data from four scans was combined to form this point cloud. c) Geometric primitives are modeled to form the BIM. This BIM was created manually by a professional service provider.

during the design phase, detection of mistakes made during construction, and simulation and space planning during the management phase (Akinci et al., 2006; GSA, 2009a,b). A BIM that is created during the design of a facility may vary significantly from the actual current condition of the facility. These differences arise from a variety of sources, such as undocumented design changes, inadvertent errors in the construction, and renovations made during the ensuing time period. Frequently, facilities have no design BIM at all, and, in some cases, even blueprints may be unavailable. Consequently, there is a substantial need to efficiently create BIMs of the "as-built" or "as-is" conditions of a facility. We use "as-is" to refer to both terms hereafter.

The geometry of as-is BIMs is starting to be generated using data from laser scanners, although image-based techniques show promise for some applications (Furukawa et al., 2009b). A laser scanner provides distance measurements of surfaces visible from the sensor's viewpoint, which can be converted to a set of 3D points known as a point cloud (Figure 1). Individual point measurements can be accurate to a few centimeters or less than a millimeter depending on the sensor, the range, and the surface being scanned. To obtain sufficient coverage of a building, the scanner is placed in various locations throughout and around the facility, and the point clouds from each location are combined together in a common coordinate system – a process known as registration. More recently, mobile scanning systems have become available, greatly reducing the scanning time at the expense of some accuracy of the registered point cloud (Trimble, 2011).

Point clouds can be used for some applications, such as clash detection (i.e., identifying collisions between built components and those in a design

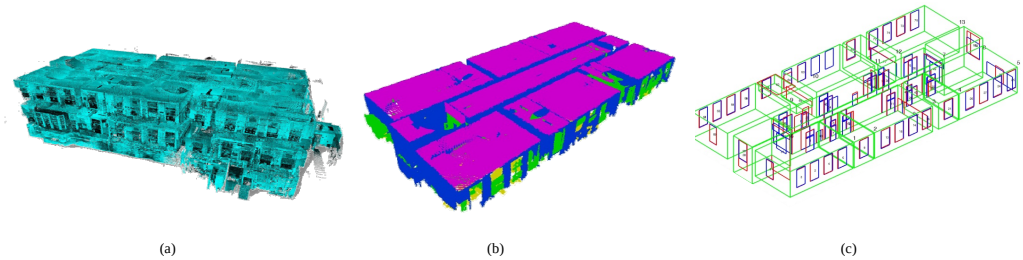
model), but for many purposes, a higher level BIM representation is advantageous. An as-is BIM allows analysis and manipulation of the model at the component level (e.g., walls and doors) rather than at the individual point level, which is more natural, efficient, and compact, since components can be summarized by a small number of parameters (Figure 1c).

The process of converting point cloud data into an as-is BIM is known as "scan-to-BIM". Geometric surfaces or volumetric primitives are fitted to the 3D point cloud to model walls, floors, ceilings, columns, beams, and other structures of interest. The modeled primitives are annotated with identity labels (e.g., wall) and meta-data, such as the surface material (e.g., concrete), and spatial and functional relationships between nearby structures and spaces are established. Currently, the scan-to-BIM process is primarily a manual operation, and it is labor-intensive and error-prone (Anil et al., 2011b). Even with training, the result produced by one modeler may differ significantly from that produced by another person. Our goal is to develop tools to help automate this process using techniques from computer vision and machine learning (Xiong and Huber, 2010; Adan et al., 2011; Adan and Huber, 2011, 2010; Anil et al., 2011a). Researchers in the AEC domain recognize the need for such automation tools (Brilakis et al., 2010; Tang et al., 2010). Our hypothesis is that the development of these tools will lead to a better understanding of how to learn and represent the fundamental principles of building design and construction.

This article describes our recent work on automatically creating as-is BIMs from laser scan data. Our method takes as input a set of registered 3D point clouds obtained from various locations in a room, and automatically

identifies and models planar walls, floors, ceilings, and any significant rectangular openings (e.g., doorways and windows). Applied to all the rooms in a building, our method will automatically produce a compact, semantically rich, 3D model that, while not strictly a BIM in the traditional sense, contains the geometric and identity information that substantially makes up the BIM. The remaining steps of labeling windows and doorways and converting the model from a surface representation to a volumetric representation are the subject of ongoing work.

One of the key challenges to automating the as-is BIM creation process is the problem of occlusions. Building modeling algorithms are frequently demonstrated on simple examples like hallways that are devoid of furniture or other objects that would obscure the surfaces to be modeled. To be practical, modeling algorithms need to function in natural, unmodified environments, since removing furniture prior to scanning usually is not feasible. Not only do occluding objects block visibility of the surfaces of interest, they may also be inadvertently interpreted as parts of the model themselves. For example, a large cabinet against a wall may look very similar to a wall. An automated modeling algorithm must be capable of reliably distinguishing between such clutter objects and the target surfaces. Our algorithm addresses the challenges of clutter and occlusion by explicitly reasoning about them throughout the process. To distinguish clutter from non-clutter, it is necessary to learn a model of what clutter looks like and how it is different from walls, ceilings, and floors. To understand occlusions, we use a ray-tracing algorithm to identify regions that are occluded from every viewpoint and to distinguish these regions from openings in the surface (e.g., due to doorways



(a) (b) (c)

Figure 2: Algorithm overview. Given a registered point cloud of a facility (a), each room is analyzed in two phases. First, context-based modeling (b) recognizes and models key structural components – walls (blue), floors (yellow), ceilings (magenta), and clutter (green). Detailed surface modeling (c) detects and models openings (red rectangles) from windows, doors, and built-in cabinets and fills in occluding regions on each wall, ceiling and floor surface.

or windows).

Broadly, our algorithm consists of two phases (Figure 2). The algorithm takes as input a set of registered scans and a model of the characteristics of openings within walls, which is learned from training examples. The output consists of labels for each of the points within the scans (wall, floor, ceiling, or clutter), labeled patches for planar regions within the point cloud, an adjacency map indicating which patches are connected to one another, and a set of openings detected within each planar patch. In the first phase, planar patches are extracted from the point cloud and a context-based machine learning algorithm is used to label the patches as wall, ceiling, floor, or clutter (Figure 2b). These patches are intersected with one another to form a simple surface-based model of the room. In the second phase, each planar

surface is analyzed to identify and model the occluded regions and openings (Figure 2c). A learning algorithm encodes the characteristics of opening shape and location, which allows the algorithm to infer the shape of an opening even when it is partially occluded. An inpainting algorithm fills in the occluded regions with a realistic surface for visualization purposes. These two phases are described in more detail in Sections 3 and 4 respectively. For efficiency, the algorithm operates independently on the data from each room. The method can handle multiple scans per room, but we assume that the scans are registered with one another. In our experiments, this registration was performed manually, but automated methods for solving this problem exist (Huber and Hebert, 2003). Our algorithm operates on planar patches because the most of the surfaces of interest are planar. The extension to non-planar surfaces is the subject of ongoing work. The algorithm also assumes that the direction of up is known. This is typically provided by scanners, since they are leveled prior to scanning. If the up direction is unknown, the orientation can be estimated using statistics of the data.

2. Related Work

Many researchers have studied the problem of reconstruction of building interiors and exteriors using laser scanner data (Frueh et al., 2005; Thrun et al., 2004; Hähnel et al., 2003; Stamos et al., 2006; Böhm et al., 2007; Böhm, 2008; Pu and Vosselman, 2009; Becker, 2009; Ripperda and Brenner, 2009; Budroni and Böhm, 2010; Budroni and Boehm, 2010). Generally, the emphasis has been on creating visually realistic models rather than geometrically accurate ones (e.g., El-Hakim et al. for indoor environments (El-Hakim

et al., 1997), and Frueh et al. for outdoor environments (Frueh et al., 2005)). Many of these algorithms extract planar patches from the data but do not explicitly recognize the identity of components, such as walls, ceilings, and floors. Thrun et al. developed a plane extraction method based on the expectation-maximization algorithm (Thrun et al., 2004), and Hahnel et al. used a plane sweep approach to find planar regions (Hähnel et al., 2003). Stamos et al. combined planar patch modeling with triangular meshes in complex areas (Stamos et al., 2006).

Context-based building modeling has been studied by several other researchers (Nüchter and Hertzberg, 2008; Nüchter et al., 2003; Pu and Vosselman, 2006; Cantzler, 2003; Rusu et al., 2008). These approaches rely on hand-coded rules, such as "walls are vertical and meet at 90° angles with floors." The constraint network introduced in (Nüchter and Hertzberg, 2008) uses Horn clauses. Such rules are usually brittle and break down when faced with noisy measurements or new environments. Our context-based modeling algorithm differs from this previous work in that our approach automatically learns which features and contextual relationships are important using training data and eliminates the need to manually create these rules. Recently, Koppula et al. have used a graphical model to represent contextual relationships for recognizing objects in indoor scenes using 3D + color data (RGBD) from a Kinect sensor (Koppula et al., 2011).

Several methods have been proposed for using laser scanners to create detailed models of walls and building façades (Böhm et al., 2007; Böhm, 2008; Pu and Vosselman, 2009; Ripperda and Brenner, 2009; Frueh et al., 2005). Windows can be detected by façade modeling methods by modeling

regions of low data density with rectangles (Böhm et al., 2007; Pu and Vosselman, 2009). Model-based approaches can also be used to predict patterns in façades using top-down processing (Becker, 2009; Ripperda and Brenner, 2009; Koutsourakis et al., 2009).

Our detailed wall modeling differs from previous work because it is specifically designed to handle cluttered environments. Most previous work assumes that occlusions are minimal or that regions occluded from one view can be observed from another viewpoint (Frueh et al., 2005; Pu and Vosselman, 2009). One approach for addressing occlusions is to identify another region that matches the occluded region and use that data to fill in the missing region (Böhm, 2008). This method will work for small occlusions under the assumption that the occluded region is part of a repeated pattern on the surface, such as one of many identical windows on a wall. Real-world environments are typically highly cluttered, with furniture occluding significant portions of most walls. It is important for an algorithm to be robust to clutter, and, therefore, we evaluate our approach in unmodified and highly cluttered environments. The general problem of reconstructing occluded surfaces in 3D is also known as "hole filling," and there are many proposed solutions (Davis et al., 2002; Salamanca et al., 2008). Surfaces in buildings often have a more constrained structure, so specialized approaches can be applied in many cases (Frueh et al., 2005; DellAcqua and Fisher, 2002; Sappa, 2002).

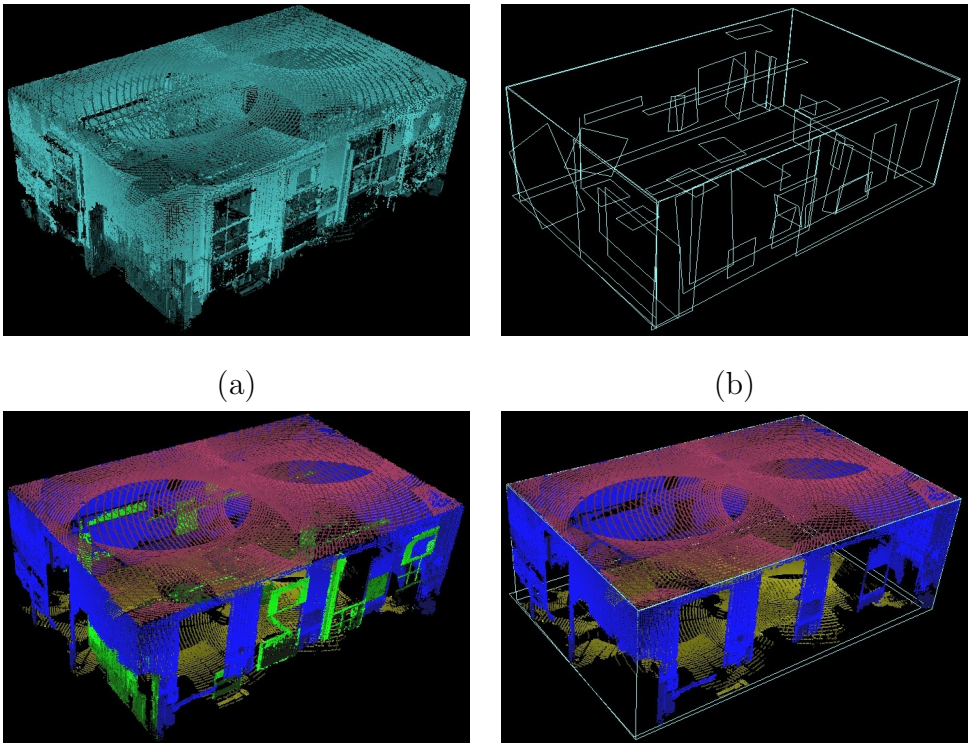
Reconstruction approaches using images or videos are also well-studied. Early work by Debevec used images to semi-automatically model building exteriors (Debevec et al., 1996). Pollefeys et al. used video to model urban environments from a moving vehicle (Pollefeys et al., 2008). More recently,

multi-view image-matching techniques have been employed to perform large-scale structure-from-motion on large image collections (Snavely et al., 2006; Agarwal et al., 2009). Dense point clouds can then be created using multi-view stereo (Furukawa et al., 2010), from which mesh-based surfaces can be constructed. So far, image-based approaches do not have the level of accuracy or data density needed for the AEC domain, but recent advances, such as Manhattan world stereo (Furukawa et al., 2009b,a), show promise.

Laser scanners and imagery have been used to address several related problems within the AEC domain. Bosché aligned laser scan data with a CAD design model to identify and measure components like steel beams and columns (Bosché, 2010; Bosche and Haas, 2008). Approaches based on (Snavely et al., 2006) have been shown to be useful for visualizing construction progress by overlaying design models with imagery (Golparvar-Fard et al., 2009). Zhu developed a method to create models from images for the special case of concrete columns using edge detection and color- and texture-based classification (Zhu and Brilakis, 2010). Previous relevant work from our group includes methods to assess the accuracy of as-is BIMs from laser scanners by analyzing patterns in the differences between the original point cloud and the reconstructed model (Anil et al., 2011b), and to compare scan data with design models for detecting defects on construction sites (Yue et al., 2006; Akinci et al., 2006).

3. Context-based Modeling

The first phase of our algorithm recognizes and models the core structural components in a room. Specifically, the algorithm recognizes planar walls,



(a) (b)

(c) (d)

Figure 3: The context-based modeling algorithm consists of four steps: a) voxelization; b) patch detection; c) patch classification; and d) patch intersection and clutter removal.

ceilings, and floors, and distinguishes these components from other surfaces, which are considered clutter in this context.

The primary challenge in the recognition of structural components is distinguishing relevant objects from clutter. The distinction can be difficult or impossible if objects are considered in isolation, especially for highly occluded data sets (e.g., a wall patch could be smaller than most clutter patches due to occlusion). Our approach is to leverage context to aid in this process. Contextual information has been shown, in other computer vision applications, to help by limiting the space of possibilities (Murphy et al., 2005) and by ensuring global consistency among multiple interacting entities (Rabinovich et al., 2007). In our situation, contextual information could help to recognize objects of interest and to distinguish them from clutter through the relationships between the target surface and other nearby surfaces. For example, if a surface is bounded on the sides by walls and is adjacent to a floor on the bottom and a ceiling on the top, it is more likely to be a wall than clutter, independently of the shape or size of that surface. In this way, the interpretation of multiple surfaces can mutually support one another to create a globally consistent labeling.

This phase of the algorithm consists of four steps (Figure 3): 1) *Voxelization.* The registered point cloud's density is reduced and distributed more uniformly; 2) *Patch Detection.* Large planar patches are extracted from the voxelized point cloud. 3) *Patch Classification.* A context-based learning algorithm uses stacking to classify the patches according to the categories wall, floor, ceiling, and clutter. 4) *Patch Intersection and Clutter Removal.* Adjacent labeled patches are intersected with one another to create more accurate

boundaries, and clutter surfaces are removed. The next several sub-sections describe these steps in more detail.

3.1. Voxelization

First, the input point cloud is discretized in a uniformly spaced 3D grid data structure, which is known as a voxel space. This discretization serves to reduce the amount of data in areas where it is overly dense, for example, on walls close to one of the sensor positions or on surfaces sensed from several locations. At the same time, sparsely sensed areas, like those seen only from a longer distance, will be maintained at their original density.

The voxelization process can be viewed as a simplified version of an evidence grid Moravec (1996). For each point, the voxel that it lies within is determined and marked as "occupied." Once all the points are added to the voxel space, the centers of the occupied voxels are used to represent the original data. Alternatively, the centroids of the points within each voxel can be used. Voxelization introduces a small amount of geometric error in the point cloud due to quantization, but can reduce the amount of point cloud data significantly. The voxel size (1.5cm) is chosen to minimize the quantization error while avoiding introducing significant numbers of holes in the surfaces.

3.2. Patch Detection

After initializing the voxel data structure, planar patches are extracted from the voxelized data. Patches are found using a region-growing algorithm to connect nearby points that have similar surface normals and that are well-described by a planar model (Figure 4). The boundary of each planar patch is

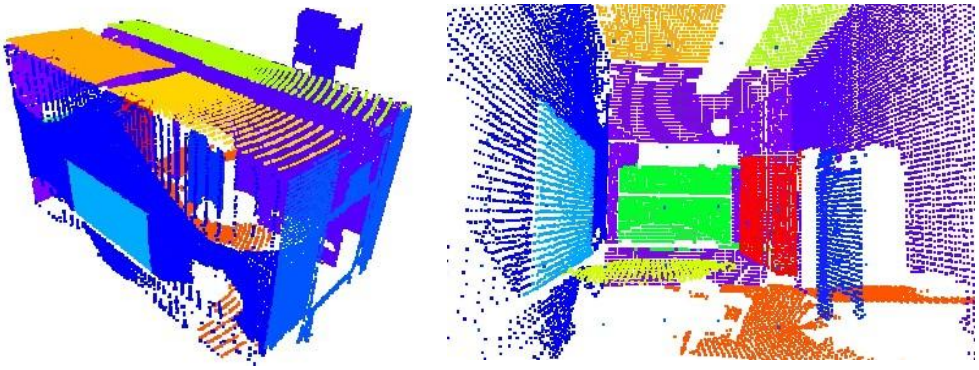


Figure 4: Patch detection. Planar patches are extracted using a region-growing algorithm. Patches for one room shown from the exterior (left) and interior (right).

described by the minimum area bounding rectangle of the constituent points.

The region-growing algorithm is similar to (Rabbani et al., 2006). For each point, the total least squares (TLS) algorithm (Golub and van Loan, 1980) is used to fit a plane to the local neighborhood of points within a specified radius (10 times the voxel size), thereby providing an estimate of the surface normal. The planarity of the point is defined as the smallest eigenvalue of the scatter matrix formed during the least squares fitting. The points are then sorted in order of decreasing planarity. The most planar point not already part of a patch is selected as a seed point and added to a new patch. Points are added to the patch as long as they are adjacent to a point already in the patch and the angle between the point's normal and the patch normal estimate is within a threshold (within 2 cm and 45° in our experiments). Once no more points can be added to the patch, all the points in the patch are used to fit a new plane to the data. This new plane is used to

check if additional points can be added to the patch using the same criteria as above. The process is repeated until no more points can be added to the patch. This iterative re-fitting of the patch plane improves the segmentation results significantly compared to the basic algorithm described in (Rabbani et al., 2006). Small patches generally correspond to non-planar regions, so once all data is assigned to a patch, only patches larger than a size threshold are kept.

3.3. Patch Classification

We begin our classification algorithm with a set of patches detected and modeled from the previous step. Then, we produce a graph by connecting each planar patch with its four nearest neighbors. Nearest neighbors are found by measuring the minimum Euclidean distance between patches' boundaries (See Figure 3(b)). We use a machine learning framework *stacking* to utilize both local and contextual information of the planar patches. Stacking was first developed by D. H. Wolpert (Wolpert, 1992). It has been successfully used in various applications, such as natural language processing (Cohen, 2005; Kou, 2007) and computer vision (Munoz et al., 2010; Xiong et al., 2011). The idea is to train a sequence of base learners used to model the underlying structure of the relational data, where each base learner is augmented by expanding an instance's feature with the previous predictions of related samples. The alternative approach is to use a probabilistic graphical model to optimize the conditional probability of labels given the data (see our previous work (Xiong and Huber, 2010)). A node in the graph is a random variable representing a patch's label and edges are formed to model context. The next subsections describe the individual components of the classification

procedure. First, we introduce the notation (Section 3.3.1) and our choice of base learner (Section 3.3.2). Then, we describe how to generate contextual features from neighboring patches (Section 3.3.3). Finally, we arrive at a stacked learning algorithm in the context of our application (Section 3.3.4).

3.3.1. Notation

We denote i^{th} patch by its feature vector $\mathbf{x}_i \in \mathcal{R}^d$ (note that this will include both local patch descriptors and contextual information), and y_i to be its ground truth label of K labels, $y_i \in \{c_1, \ldots, c_K\}$ and $\widehat{y}_i$ to denote the estimation or prediction of this patch. In this problem $K = 4$, and the possible labels are $wall, floor, ceiling$, and $clutter$. A labeled sample is a pair of $(\mathbf{x}_i, y_i)$, and a data set $\mathcal{D}$ is set of labeled samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. We use upper case letters for random variables, lower case letters for concrete assignments to these variables, and boldface for vectors.

3.3.2. Base Learner

One advantage of our approach is the free choice of base learner, which can be varied among different applications. We choose base learner $\mathcal{L}$ to be a simple K-class logistic regression (LogR) model, parameterized by weights $\mathbf{w} \in R^{d \times K}$. The conditional distribution of the i^{th} patch's label given its features is,

$$p(y_i = k|\mathbf{x}_i; \mathbf{w}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{\sum_{j=1}^K \exp(\mathbf{w}_j^T \mathbf{x}_i)}. \tag{1}$$

We train the model by maximizing the likelihood of the data:

$$\arg\max_{\mathbf{w}} \sum_i \log p(y_i|\mathbf{x}_i) - \lambda||\mathbf{w}||^2 \tag{2}$$

where $\lambda > 0$ regularizes the model. The objective is concave under $\mathbf{w}$, which implies that it has a global maximum. We find the optimal value for $\mathbf{w}$ using stochastic gradient ascent (Bertsekas et al. (2003)).

Given a training set $\mathcal{D}$, we denote $\mathbf{w} = \mathcal{L}(\mathcal{D})$ as solving the maximization problem in Equation 1. Given a patch $\mathbf{x}_i$ and parameters $\mathbf{w}$, we denote $\widehat{y}_i = f(\mathbf{x}_i, \mathbf{w})$ as predicting the label (Equation 1) for this sample.

3.3.3. Contextual Features

For modeling contextual features, we define R pairwise binary relations between two planar patches, e.g., such relations could be whether two patches are parallel/orthogonal to each other. We encode a patch's neighborhood configuration with a matrix $\mathbf{H}$, where the entry in the k^{th} row and r^{th} column, $\mathbf{H}_{k,r}$ is 1 if $y_i = c_k$ and r^{th} relationship is satisfied between itself and its neighboring patch. Otherwise, $\mathbf{H}_{k,r} = 0$. Then $\mathbf{H}$ is converted into a vector $\mathbf{h}$ by concatenating its columns. For each patch we compute such configuration between itself and each of its neighbors. The final $\mathbf{h}$ is computed as the sum of them. This vector is used for constructing an extended data set that incorporates the contextual information (Section 3.3.4). The choice of a patch's local features is described in Section 5.1.

3.3.4. Stacked Learning

Now, we describe the core of the learning algorithm: how to train a sequence of learners so as make it aware of the labels of nearby examples. By sequentially training a series of classifiers, we can ideally learn how to fix the mistake from the previous one. In addition, we can use these previous predictions as contextual cues. That is, given a labeled training set $\mathcal{D}$, we first

train a classifier $\mathbf{w}^0 = \mathcal{L}(\mathcal{D})$ over the entire training set (note that currently $\mathcal{D}$ contains patches' local features only). Using $\mathbf{w}^0$, we can classify each patch $\mathbf{x}_i$ to generate predictions $\widehat{y}_i = f(\mathbf{x}_i, \mathbf{w}^0)$ from which to derive contextual cues $\mathbf{h}_i$, as described in Section 3.3.3, and then train a new classifier $\mathbf{w}^1$. Specifically, we use each $\widehat{y}_i$ to create an extended data set with contextual cues $\mathcal{D}^1 = \{((\mathbf{x}_i, \mathbf{h}_i), y_i)\}_{i=1}^n$ and train a new classifier $\mathbf{w}^1 = \mathcal{L}(\mathcal{D}^1)$. The process can be repeated for multiple rounds until no improvement is observed.

However, note that if we were to use $\mathbf{w}^0$ to classify our training data, the resulting predictions would be more optimistically correct than it would be on the unseen test data. Instead of training a single classifier over the entire labeled set, we generate multiple temporary classifiers that are trained on subsets of the whole data. The purpose is to generate predictions over the examples that were not used to train a classifier. Such technique is known as *stacking* (Wolpert, 1992). By performing stacking, we simulate the same prediction procedure for labeled data as in predicting the unlabeled data. The details of the above procedure can be found in Algorithm 2. Finally, we end up with learning and inference methods in Algorithm 1.

After the classification, we merge coplanar and adjacent patches that belong to the same class.

3.4. Patch Intersection and Clutter Removal

The initial boundary estimation may not be accurate due to the highly-cluttered environment. Once each patch is assigned to the one of the following four labels: *wall*, *ceiling*, *floor*, and *clutter*, we re-estimate the patch boundary by intersecting wall patches with neighboring walls, floor, and ceiling. It is using the building structural constraint that the walls, ceiling and

Algorithm 1 Stacked Learning

Inputs: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$.

Learning algorithm:

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^l$.

Let $\mathcal{D}^0 = \mathcal{D}$ and train a local model $\mathbf{w}^0 = \mathcal{L}(D^0)$.

for $t = 1 \ldots T$, we train the stacked models as follows:

1. Apply procedure `hold-out prediction` described in Algorithm 2 to labeled data and obtain predictions $\hat{\mathbf{y}}_{1,\ldots,l}$.
2. Construct an extended data set $\mathcal{D}^t$ by performing a feature expansion as follows: $\mathbf{x}_i^t = (\mathbf{x}_i, \mathbf{h}_i)$.
3. Train model $\mathbf{w}^t = \mathcal{L}(\mathcal{D}^t)$.

return learned models $\mathbf{w}^{0,\ldots,T}$.

Inference algorithm: given test data $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$

$\widehat{\mathbf{y}}^0 = f(\mathcal{X}, \mathbf{w}^0)$.

for $t = 1 \ldots T$

1. carry out step 2 (in Learning algorithm) to produce $\mathcal{X}^t$.
2. $\widehat{\mathbf{y}}^t = f(\mathcal{X}^t, \mathbf{w}^t)$.

return $\widehat{\mathbf{y}}^T$.

Algorithm 2 Hold-out Prediction

Inputs: labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{l=1}^{n}$, base learner $\mathcal{L}$.

1. Split $\mathcal{D}$ into J equal-sized disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_J$
2. Train a temporary $\mathbf{w}_j$ for each subset $\mathcal{D}_j$, where $\mathbf{w}_j = \mathcal{L}(\mathcal{D} - \mathcal{D}_j)$.
3. Use temporary $\mathbf{w}_j$ to predict on subset $\mathcal{D}_j$. Obtain prediction $\widehat{\mathbf{y}} = f_j(\mathbf{x}, \mathbf{w}_j), \mathbf{x} \in \mathcal{D}_j$.

return $\widehat{y}_{1,\ldots,n}$.

floor define an enclosed space for an indoor room.

4. Detailed Surface Modeling

The surfaces produced by context-based modeling represent idealized surfaces that are perfectly planar and unoccluded and with no openings. Real surfaces in unmodified environments, particularly walls, are usually heavily occluded, and it is important to understand where those occlusions occur. For example, if an engineer wants to measure a dimension of a room, he or she would likely prefer to make the measurement at a location that was actually observed, rather than at an occluded position, which might be incorrectly modeled. Furthermore, occlusions increase the challenge of estimating the boundaries of openings, such as doorways and windows.

The detailed surface modeling algorithm operates on each planar patch S_i produced by the context-based modeling process, detecting the occluded regions and regions within openings in the surface. The process involves three steps for each patch. 1) *Occlusion Labeling.* Ray tracing is used to determine which surface regions are observed, which are occluded, and which

are empty space (Section 4.1). 2) *Opening Detection.* A learning-based method is used to recognize and model openings in the surface based on the occlusion labeling results (Section 4.2). 3) *Occlusion Reconstruction.* Occluded regions not within an opening are reconstructed using a 3D hole filling algorithm (Section 4.3). The next several subsections describe these steps in detail.

4.1. Occlusion Labeling

Occlusion labeling uses a voxel space similar to that used in the context-based modeling algorithm. However, in this case, the axes of the voxel space are aligned with the axes of the patch with the plane of the patch passing through the center of one layer of voxels. In practice, only this single layer of voxels is used. In this step, each voxel on the surface S_j is given one of three labels: occupied (F – for full), empty (E), or occluded (O). The labeling is performed once for each surface, and the information is then combined into a unified labeling (Figure 5).

First, the points from the set $\mathbb{O} = \{O_1, O_2, \ldots O_K\}$ of all scans from which the surface S_j is visible are inserted into the voxel space. Each voxel lying on the surface is labeled as occupied (F) if any point falls into the voxel, or empty (E) otherwise. We denote this labeling L_0.

Given a labeling L_0 for surface S_j, it is impossible to differentiate between a voxel that truly represents free space and one that is just occluded. We address this problem using a ray-tracing algorithm that explicitly reasons about occlusions between the sensor and the surface (Figure 5). A separate voxel labeling L_k is created for each scan O_k in the set $\mathbb{O}$. The labeling is initialized to L_0. A ray is traced from the scanner origin to each measured

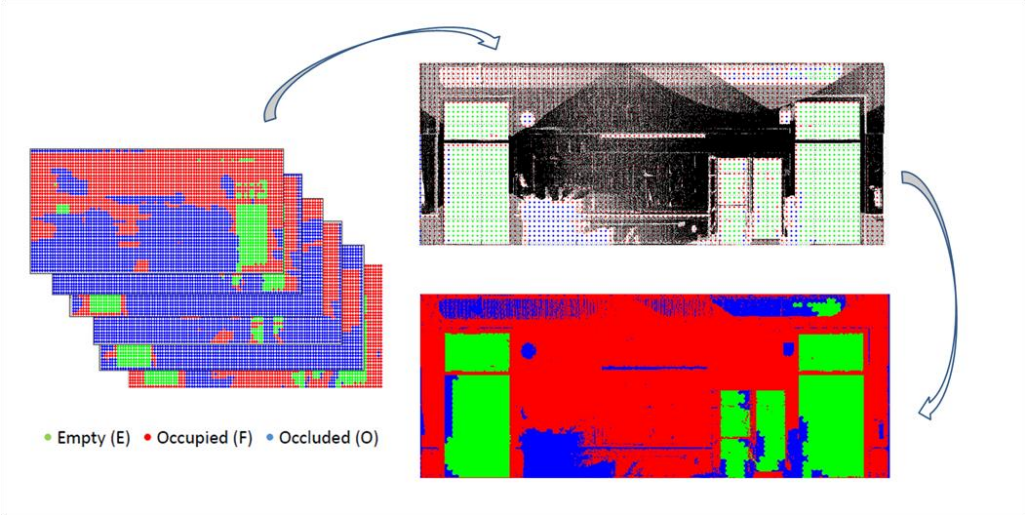


Figure 5: Occlusion labeling. Ray-tracing is used to label each surface as occupied (red), occluded (blue), or empty (green) from each viewpoint that observes a surface (left). The labels from all viewpoints are merged into a single representation (top right) and a region-growing algorithm labels the unobserved regions (bottom right).

point in O_k. For efficiency, only those points lying within the bounds of the patch boundary need to be considered. If the point lies beyond the surface, the voxel lying at the intersection of the ray and the surface is labeled empty (E) since the sensor observed a point further than the surface in that direction. Similarly, if the point lies closer to the sensor than the surface, then the voxel at the intersection is labeled occluded (O).

This ray-tracing process produces K labels for each voxel on the surface. The single scan labels L_k are combined with the initial labeling L_0 to form the final labeling L_F by initializing L_F to L_0 and applying the following rule to each voxel v in S_j:

$$\text{If } L_0(v) = E \text{ and } L_j(v) = O, \forall j = 1, 2 \ldots K,$$
$$\text{then } L_F(v) = O \tag{3}$$

The v argument of a labeling indicates the labeling of voxel v.

Voxel-based processing is well-suited for ray-tracing of very large data sets, but for detailed surface modeling, we want the use the highest resolution possible. Therefore, we convert the voxel labeling into a high-resolution, image-based representation. The voxel size is approximately $1/25^{\text{th}}$ of the resolution of the original data. To recover a high-resolution representation, we use the voxel centers as seeds to a region growing algorithm that fills the gaps where no data was observed.

Each surface S_j is represented by a 2D image I_j which is mapped to the surface with the image axes aligned with the voxel space axes. The size of the image is determined by the surface boundary shape and the scale of the image. The pixels in the image correspond to a physical size and location on the wall. The image is essentially a texture map.

The pixels in I_j are initialized with an "unknown" label (U). The point measurements that fall within the surface voxel space are projected orthographically onto the surface plane, and the corresponding pixels in I_j are labeled occupied (F). The centers for voxels labeled empty or occluded in L_F are projected in the same fashion, and the corresponding pixels are labeled E and F respectively.

The region growing algorithm is designed to infer the labels of unknown pixels that may be empty or occluded. The algorithm operates iteratively, beginning with $I_j^0 = I_j$, where the superscript t indicates the iteration number. The following rules are applied to each unknown pixel in the current iteration. The j subscript is omitted for clarity. The first rule grows the occluded regions.

$$I^{t+1} = O \text{ if } I^t(x,y) = U \text{ and } d_{min}(I^t(x,y), \mathbb{I}_E^t) > \alpha \\ \text{and } d_{min}(I^t(x,y), \mathbb{I}_O^t) < \beta \tag{4}$$

where $\mathbb{I}_E^t$ and $\mathbb{I}_O^t$ are the sets of pixels in I^t labeled E and O respectively and $d_{min}(a, \mathbb{B})$ is the minimum Euclidean distance between the pixel a and any pixel in the set $\mathbb{B}$. When no more pixels are changed in an iteration, the second rule, which grows empty regions, is applied.

$$I^{t+1} = E \text{ if } I^t(x,y) = U \text{ and } d_{min}(I^t(x,y), \mathbb{I}_O^t) > \alpha \\ \text{and } d_{min}(I^t(x,y), \mathbb{I}_E^t) < \beta, \tag{5}$$

The algorithm terminates when no pixels are changed in an iteration. The final result of occlusion labeling is a labeled image I_j with no unknown pixels (Figure 5).

The occlusion labeling process is summarized in Algorithm 3. The occupancy subroutine updates the occupancy label for the voxel containing p subject to the constraints of S_j. The ray-trace subroutine updates the labeling L_k according to the intersection of the ray to p with S_j.

Algorithm 3 Occlusion Labeling

Inputs: surface S_j, scans $\mathbb{O} = \{O_1, O_2, \ldots O_K\}$.

Output: labeled image I_j.

for all points $p \in \mathbb{O}$ **do**

occupancy$(p, S_j) \rightarrow L_0$

end for

for all scans $\mathbb{O}_k \in \mathbb{O}$ **do**

$L_k = L_0$

for all points $p \in \mathbb{O}_k$ **do**

ray-trace$(p, S_j) \rightarrow L_k$

end for

end for

$L_F =$ merge-labels$(L_0 \ldots L_k)$ (Equation 3)

$I_j =$ region-grow(L_F) (Equations 4 and 5)

4.2. Opening Detection and Modeling

In the ideal case, an opening in a surface could be extracted directly from regions in I_j that are labeled "empty." However, in real situations, openings are rarely so straightforward. They may be partially occluded by furniture or other objects, in which case the opening boundary must be inferred (Figure 6a-b). Openings may also contain occupied regions, for example, due to

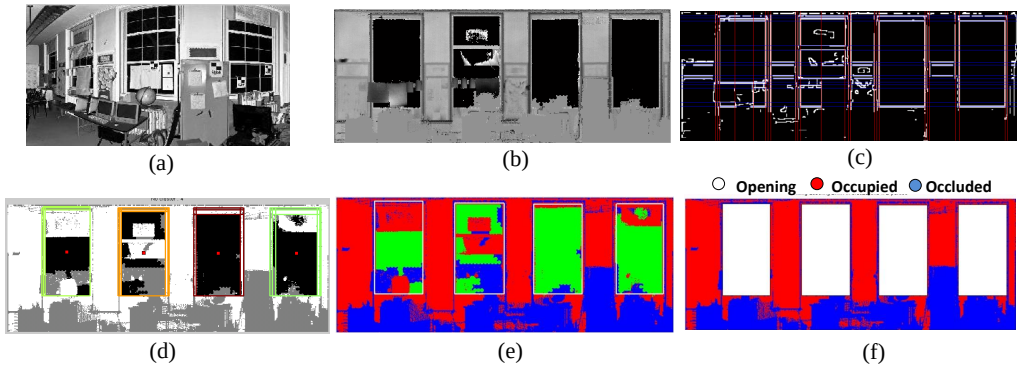


Figure 6: Opening detection. (a) Reflectance image of a wall surface. (b) Corresponding depth image. (c) Detected lines overlaid on edge image. (d) Openings detected by the SVM-based detector, with each cluster of openings in a different color. (e) Prototype openings after clustering, superimposed on I_F. (f) Final labeling with openings masked out. (Figure reprinted from Adan and Huber (2011))

a window-mounted air-conditioner. We overcome the challenge of modeling partially occluded or occupied openings by learning a model of the typical size, shape, and location of openings from training examples. In regions where information about the boundary is unavailable, the learned model provides guidance based on other, similar situations where the information was available. Our algorithm detects and models rectangular-shaped openings, which is the predominant shape in most buildings. It uses features computed from the occlusion labels and from depth edges to learn a model of openings using a support vector machine (SVM) classifier.

The classifier uses a set of fourteen features for deciding whether a hypothesized opening is an actual opening or not. An opening hypothesis Θ is a rectangular region in I_j with width w and height h located in a surface of width W and height H. The features for an opening hypothesis are as follows: 1) area (wh); 2) aspect ratio (w/h); 3-4) size relative to encompassing surface (w/W and h/H); 5-8) distances from the sides of Θ to the edges of the surface; 9) residual of TLS plane fit of points inside Θ; 10-12) percentages of pixels within Θ labeled E, F, and O; 13) number of hypotheses fully contained within Θ; and 14) number of inverted U-shapes (i.e., rectangles located at the bottom of the wall). Feature 13 focuses on hypotheses that belong to an outer window frame that contains hypotheses for individual window panes. Feature 14 focuses on doors, which often contain inverted U-shapes in their interior frames.

For training, the algorithm uses features computed from a set of manually labeled examples to learn an SVM with a radial basis function (RBF) kernel. At run time, we create opening hypotheses from depth edges extracted from a

range image. The range image is created by projecting points within a given distance of the surface S_j onto the surface. We use the Canny algorithm to find edges in the resulting range image and then find strong horizontal and vertical lines using a Hough transform (Figure 6c). This process produces a relatively small number of potential opening boundaries, and it is feasible to exhaustively enumerate and evaluate opening hypotheses. A hypothesis is eliminated if it does not have sufficient support from the edge image in terms of the percentage of the boundary that contains detected edges.

The result of the classifier is a set of opening hypotheses that are considered to be actual openings. In practice, each opening may be covered by several overlapping detections (Figure 6d). We group overlapping detections using k-means clustering. The value of k is chosen by minimizing the total variance within clusters and maximizing the distance between clusters. The final boundary of each opening is computed from the average of the boundary positions of the hypotheses within each cluster (Figure 6e-f).

The opening detection process is summarized by Algorithm 4. The subroutine range-image computes a range image for the points near S_j. Edge-detect performs Canny edge detection on an image. The hough-transform-filter routine uses the Hough transform to find vertical and horizontal edges from an image. The rectangle-hypotheses subroutine enumerates the potential rectangles within a set of horizontal and vertical lines. The classify subroutine computes the feature vector for a hypothesis Θ and performs SVM classification on the result using the supplied opening model. Finally, the cluster subroutine clusters overlapping hypotheses.

Algorithm 4 Opening Detection

Inputs: surface S_j, scans $\mathbb{O}$, labeled image I_j, opening model M.

Output: opening hypotheses $\mathbb{H}_2$.

$R = \text{range-image}(\mathbb{O}, S_j)$

$E = \text{edge-detect}(R)$

$E_{HV} = \text{hough-transform-filter}(E)$

$\mathbb{H}_0 = \text{rectangle-hypotheses}(E_{HV})$

$\mathbb{H}_1 = \Theta \in \mathbb{H}_0 | \text{classify}(\Theta, M) > t$

$\mathbb{H}_2 = \text{cluster}(\mathbb{H}_1)$

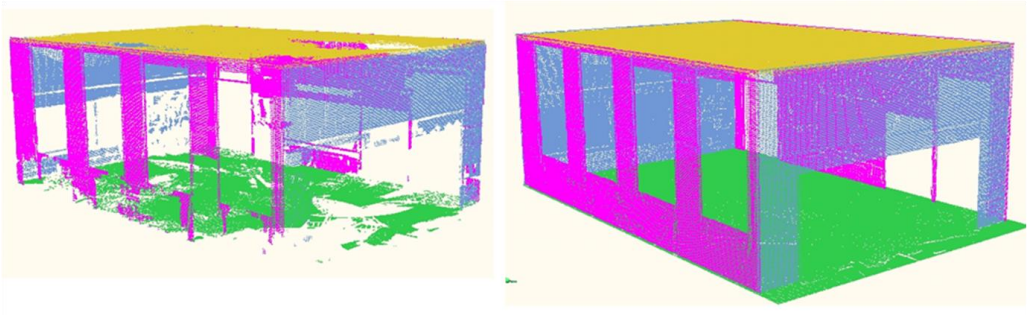


Figure 7: Occlusion reconstruction. The point cloud for one room before reconstruction (left) and afterward (right).

4.3. Occlusion Reconstruction

Once the openings are determined, the occluded regions of the wall are filled in with an inpainting algorithm (Figure 7). Inpainting is traditionally used to recover missing regions of damaged photographs (Bertalmio et al., 2000). The same idea can be extended to 3D to recover missing surfaces in occluded regions. This step is not strictly necessary, but it improves the visualization of the results. We use a version of the 3D gap-filling technique proposed by Salamanca et al. (Salamanca et al., 2008), which is itself a 3D extension of the image-based Markov Random Field inpainting algorithm proposed by Roth et al. (Roth and Black, 2005). The method operates on the 2D range image created in the previous step. Regions within detected openings are masked out, and a median filter is applied to the remaining pixels to remove range spikes that sometimes occur at the depths boundaries, which are due to the mixed pixel effect (Tang et al., 2007). For aesthetic purposes, the reconstructed regions are degraded using Gaussian noise of the same magnitude as found in the sensed data. Finally, the range image is converted back to a 3D mesh for visualization by triangulating adjacent pixels.

5. Experimental Results

We conducted experiments using data from a building that was manually modeled by a professional laser scanning service provider. The facility is a two-story schoolhouse containing 40 rooms. The facility was scanned from 225 locations resulting in over 3 billion 3D measurements.

5.1. Patch Classification

In order to handle the large data sets, each scan was sub-sampled by a factor of 13. For evaluation, the ground truth planar patch labels were derived from the professionally created model. We label each point in the input data according to the label of the closest surface in the overlaid model, and then label each patch by the majority vote of the points it contains. Furthermore, we manually validated the correctness of these labelings.

In our experiments, we split the data into two sets. We use all 23 rooms on the 1st floor for training and validation and all 13 rooms on the 2nd floor for testing. We perform classification on a per-room basis and use leave-one-out cross-validation to choose the regularization parameter λ. Our training set contains 780 planar patches, while the 13 testing rooms contain 581 patches.

Below are the features we considered for the experiments. The local features we used are patch's orientation (angle between its normal and z-axis), area of bounding rectangle, height (max z value), point density, and aspect ratio (between length and width of the bounding rectangle). Features related to area and height are normalized on a per-room basis because the size

	clutter	wall	ceiling	floor		clutter	wall	ceiling	floor
	0.87	0.84	**1.00**	0.87		0.95	**0.66**	**0.82**	0.81
P	0.86	0.82	0.93	**0.93**	R	0.95	0.60	**0.82**	0.81
	0.81	**0.88**	0.92	0.82		**0.97**	0.41	0.71	**0.88**

Table 1: Precision (P) and recall (R) for individual classes from our stacking algorithm (top row), the conditional random field algorithm, and the LogR method (bottom row).

and height of patches in one class may vary from room to room, but the relative values comparing with largest instance of the room tend to be more consistent. Local features are chosen to be independent of each other. Also they are chosen to be discriminative so that patches from different classes are separated as far as possible in the feature space. The pairwise relations we considered are *orthogonal*, *parallel*, *adjacent*, and *coplanar*. During the cross-validation, we compare different combinations of contextual features and find out coplanar+orthogonal yields the highest validation accuracy. Note that the best combination is **not** the one with all the relations.

We compare with the conditional random field (CRF) approach (Xiong and Huber, 2010) as well as our base learner LogR trained using only local patch features. All three algorithms use the same training, validation, and testing sets as well as the same features described above (except LogR cannot use the contextual features). To compare the results from all algorithms we consider the average of the per-class F_1 scores. The F_1 score of a class k is the harmonic means of its precision p_k and recall r_k and is defined as $2p_k r_k/(p_k + r_k)$. Our method achieved an average F_1 score of 0.85 over 4 classes while CRF and LogR average 0.83 and 0.78, respectively. Table I shows the individual class performance of the three algorithms. The highest confusion is between wall and clutter objects. This is because many wall and clutter patches possess similar local features due to occlusion. Our method achieves much higher recall on walls than the local model, which verifies our hypothesis that context can guide the classification procedure by ensuring neighboring consistency even when using local features alone is impossible to distinguish between the two.

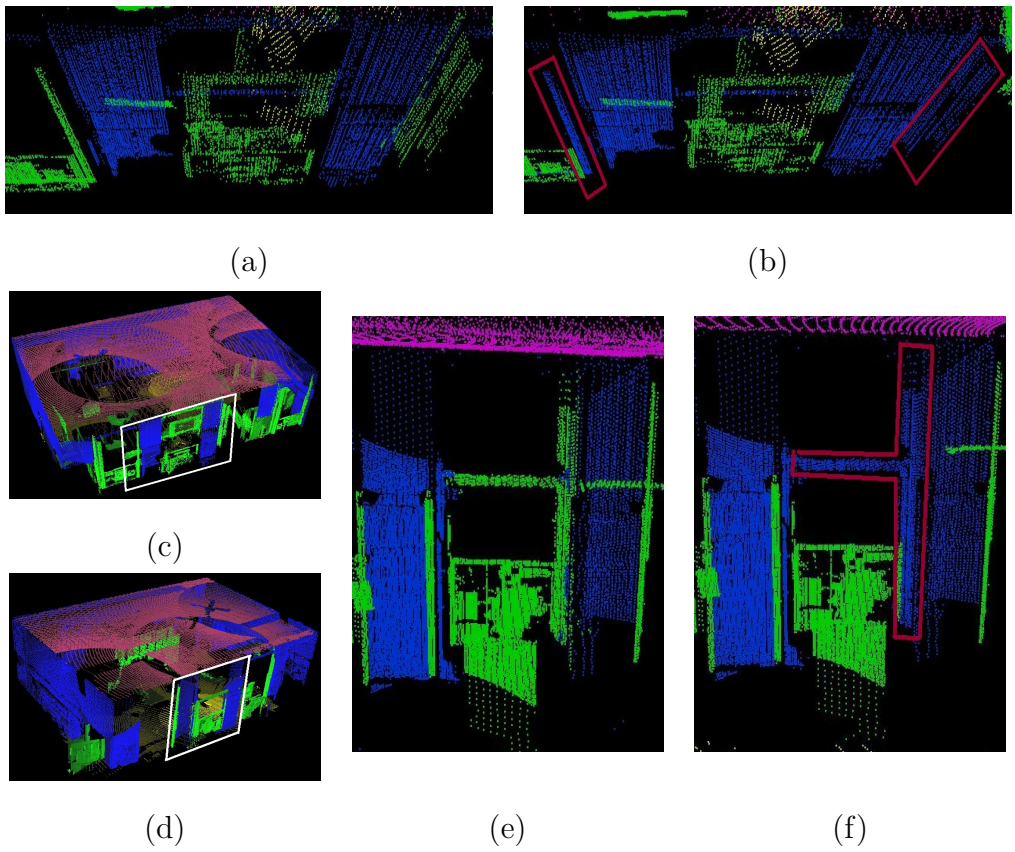


Figure 8: Comparison of results for the stacking algorithm and the LogR method. (a) A close-up of our method's output from the room shown in (c). The LogR's output on the same region is shown in (b). The mislabeling of two clutter patches are corrected by considering the orthogonality between them and their nearby wall patches. (e) and (f) show a close-up of our method's and LogR's outputs from another room shown in (d). The LogR's misclassification is fixed by considering it being coplanar with a nearby clutter object. The errors are highlighted in red and regions of focus are highlighted in white. Point coloring legend: walls (blue), floors (yellow), ceilings (magenta), and clutter (green).

Figure 8 illustrates how coplanar and orthogonal relations help to correct the misclassification made by the LogR method. The model learns coplanar objects are likely to belong to the same category and walls are prone to be orthogonal to each other. Observing two nearby patches being orthogonal increases the probability of them being wall. Figure 9 shows some detailed results viewed from the interior of three different rooms. Our algorithm successfully distinguish wall-like clutter objects (e.g. cabinets) from walls. The complete output on the 2$^{\text{nd}}$ floor can be found in Figure 10 (a). The main failures occur in the interiors of low built-in closets and in stairwells (Figure 10b), both of which are atypical situations.

5.2. Detailed Surface Modeling

We conducted two sets of experiments for the detailed wall modeling. The first evaluation used surfaces produced by a stand-alone wall detection algorithm described in Adan and Huber (2011). The second evaluation used the surfaces produced by the context-based modeling algorithm. Our results for both experiments are similar, but the specifics of which surfaces were evaluated differ somewhat.

Our algorithm has a few free parameters, which were set by evaluation on an independent subset of the data. The Hough transform threshold for the number of edge point votes needed for line detection was set to 90, and the edge support threshold t_{supp} was set to 50%. The thresholds α and β were 1 and 2 times the voxel size respectively.

The SVM classifier comprising the opening detector was trained on an independent set of 370 examples (split 50/50 between openings and non-openings). This data was further partitioned into a training set (95%) and

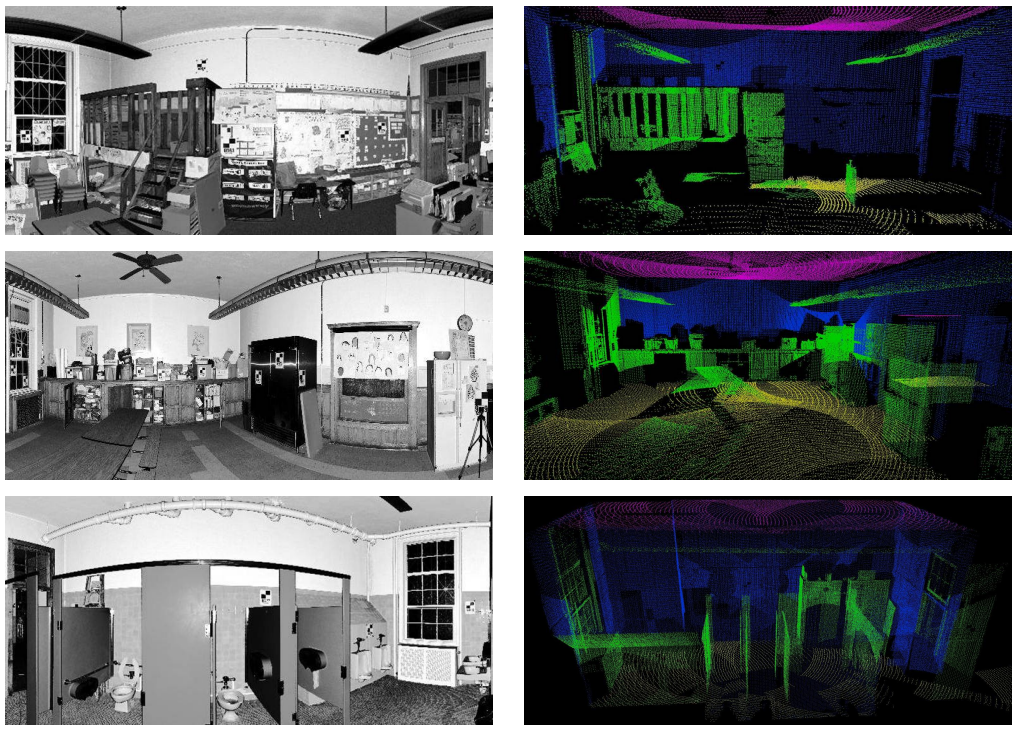


Figure 9: More results. The left column shows the reflectance images of three different rooms from our data set. The right column shows the classification results from our algorithm from the same viewpoint. Point coloring legend: walls (blue), floors (yellow), ceilings (magenta), and clutter (green).

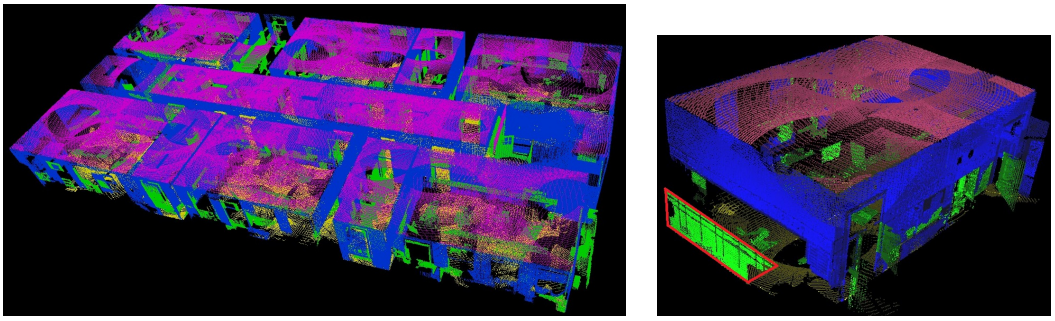


Figure 10: The classification result for the entire second floor is shown on the left (10 cm voxels). On the right, an example of misclassification (highlighted by red) made by our algorithm.

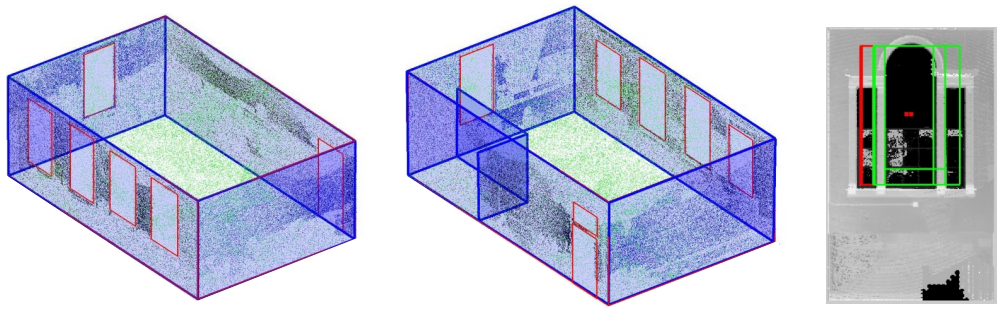


Figure 11: Example results of the detailed surface modeling algorithm for entire rooms. a) The room shown in Figure 2. Walls, ceilings, and floors are outlined in blue, and windows and doors are outlined in red. b) Another room. c) An example where the opening detection failed due to an arched window, which violates the assumptions of the algorithm.

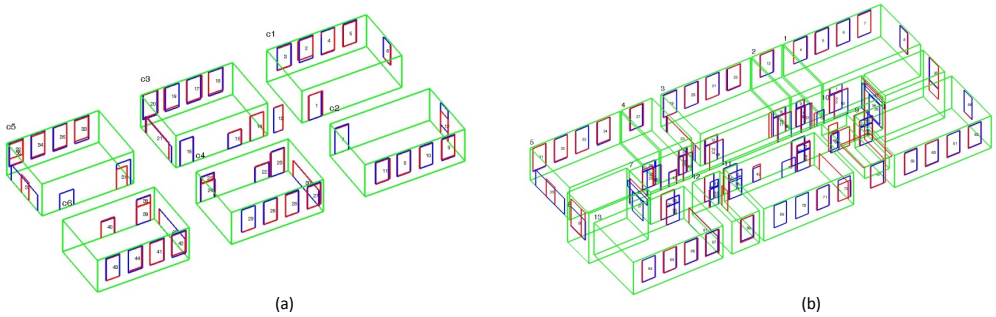


Figure 12: Example results for all rooms on the second floor. (a) First evaluation on 6 rooms. (b) Second evaluation on all 13 rooms. Detected openings are shown in red. Ground truth is overlaid in blue. Wall patches are shown in green.

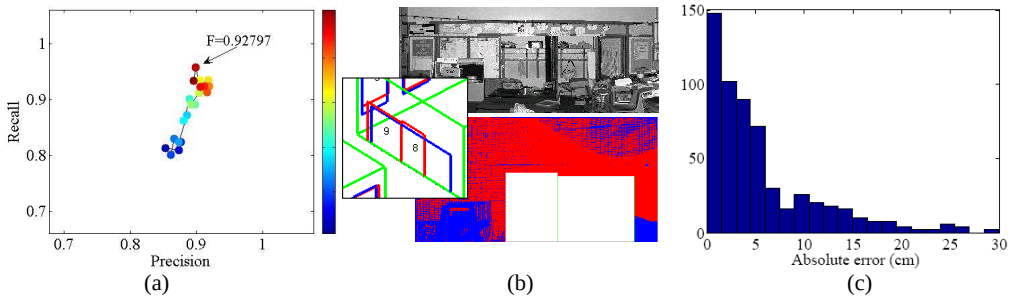


Figure 13: (a) Precision-recall curve for the opening detector performance with 10 cm voxels. (b) The detector failed on several instances, such as this one, where some doors of the closet were not open during data collection. (c) The histogram of the magnitudes of errors in opening boundary positions.

validation set (5%) for determining the classifier parameters and preventing overfitting.

In the first experiments, we evaluated the algorithm using 10 of the 40 rooms (4 from the 1st floor and 6 from the 2nd). This data set focuses mainly on the large and simple rectangular rooms in the building. Figure 12a shows the results for the 2nd floor rooms. The walls in this data were challenging due to clutter, with 35% of the total wall area occluded, 15% within an opening, and the remaining 50% unoccluded.

We focused most of our analysis on understanding the performance of the opening detection and modeling steps, since this aspect of the algorithm is considerably less studied than planar wall modeling. We evaluated two voxel sizes: 10 cm and 5 cm. The smaller size approached the limit of memory usage in our implementation, and smaller voxel sizes also increase the risk of missing wall sections due to the fact that most walls are not perfectly planar. The resolution of the images I_j was set to 2 cm/pixel, which is the resolution of the final reconstruction.

We considered two aspects of the performance. First, how reliably can the openings be detected, and second, among the detected openings, how accurately are they modeled? To answer the first question, we compared the detected openings with openings in the ground truth model (i.e., doorways, windows, and closets). The precision-recall curves for the detector, 10 cm voxel resolution, are shown in Figure 13a. We compute the best threshold using the F-measure ($F = 2PR/(P+R)$, where P = precision and R = recall). At this threshold, the algorithm correctly detects 93.3% of the openings (70 out of 75) with 10 cm voxels, and 91.8% with 5 cm voxels. Failed detections

mainly occur in regions of severe occlusion and in closets for which the doors were closed during data collection (Figure 13b).

We addressed the second question, regarding the accuracy of the reconstructed openings, by comparing the ground truth positions of the sides of each opening with the positions estimated by our algorithm. We define the absolute error for modeling one side of an opening as the absolute magnitude of the difference between the ground truth and modeled position of that edge in the direction normal to the ground truth edge and also within the plane of the modeled surface. The relative error normalizes the absolute error by the ground truth width (for the left and right sides) or height (for the top and bottom) of the opening. The average absolute error was 5.39 cm with a standard deviation of 5.70 cm, whereas the average relative error was 2.56%.

The second set of experiments was evaluated using all 13 rooms on the second floor (Figure 12b). In this case, the recognition rate was 88% (61/69), and the average error in opening size was 4.35 cm. This performance is slightly worse in recognition rate when compared to the first experiments, but the set of rooms and openings is more challenging in this case, since we include all rooms, including hallways and stairwells.

The detailed wall modeling algorithm worked well despite the high levels of occlusion and missing data in the openings. The opening detection performance was good, and the detection failures were reasonable for the given data. The accuracy of the boundaries of the openings is good, but could be improved. On our evaluation data, 36% of the boundaries fall within 2.5 cm of the ground truth. For comparison, the General Services Administration, which has established guidelines for BIM accuracy, uses an error tolerance

of 5 cm for "Level 1" and 1.3 cm for "Level 2"GSA (2009a). We anticipate that higher accuracy would be achieved using the full resolution point cloud. The final reconstructions, with the occluded areas filled in, are visually consistent with the manually created ground truth model. We also evaluated the algorithm on the easier problem of floor and ceiling reconstruction, and no openings were detected in these surfaces (which was the correct result).

6. Future Work

We are working on several improvements and advancements to our automatic building modeling algorithms. First, we are working on completing the scan-to-BIM pipeline. The model that we create is a surface-based model, whereas BIMs are typically represented using volumetric primitives. The problem of converting surface models to volumetric models has been studied in the context of reverse engineering machined parts, but is not well-explored for the problem of building modeling. We are currently developing an algorithm to automatically translate the surface-based model into a volumetric model and export it in IFC format, which is a standard file format for representing BIMs.

The context-based modeling methods work well for our case study, which contains a wide variety of rooms and geometries, but we want to evaluate on a more diverse set of environments. Another possible advancement is to integrate the opening detection and modeling algorithm directly into the context-based recognition algorithm. A tighter integration would allow us to more effectively leverage contextual relationships between openings and walls. For example, the fact that a surface contains a door-like opening would

support the interpretation of the surface as a wall. The current approach requires commitment to the wall class before openings are detected. Another potential improvement is to utilize relationships among multiple rooms. If a surface in one room is likely to be a wall, then a nearby parallel surface in the adjacent room with opposite surface normal is likely to be the other side of the wall. Such an approach could help in situations where one surface is difficult to recognize due to clutter or occlusions.

We are also investigating ways to improve the detailed surface modeling algorithm. First, we are looking to improve the boundary accuracy, which may be limited by the simple opening model that we use. Windows and doorways are normally surrounded by moldings and casings that exhibit typical geometries. Laser scanners also often sense the muntins and rails that divide windows into panes. These patterns can confuse the current algorithm because it does not understand such features. Along these lines, we have preliminary work on modeling linear moldings surrounding doorways and windows (Valero et al., 2011). We are also considering ways to incorporate color and laser reflectivity information into the modeling algorithms.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0856558 and by the Pennsylvania Infrastructure Technology Alliance. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We thank Quantapoint, Inc., for providing experimental data.

References

Adan, A., Huber, D., 2010. Reconstruction of Wall Surfaces Under Occlusion and Clutter in 3D Indoor Environments. Tech. Rep. CMU-RI-TR-10-12, Pittsburgh, PA.

Adan, A., Huber, D., 2011. 3D Reconstruction of Interior Wall Surfaces Under Occlusion and Clutter. In: Proceedings of 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT). Hangzhou, China.
URL http://www.ri.cmu.edu/pub_files/2011/5/3dimpvt-final.pdf

Adan, A., Xiong, X., Akinci, B., Huber, D., 2011. Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data. In: Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC).

Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., Szeliski, R., Sep. 2009. Building Rome in a Day. 2009 IEEE 12th International Conference on Computer Vision, 72–79.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459148

Akinci, B., Boukamp, F., Gordon, C., Huber, D., Lyons, C., Park, K., 2006. A Formalism for Utilization of Sensor Systems and Integrated Project Models for Active Construction Quality Control. Automation in Construction 15 (2), 124–138.

Anil, E., Akinci, B., Huber, D., 2011a. Representation Requirements of As-is Building Information Models Generated from Laser Scanned Point Cloud Data. In: Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC). Seoul, Korea.

Anil, E., Tang, P., Akinci, B., Huber, D., 2011b. Assessment of Quality of As-is Building Information Models Generated from Point Clouds Using Deviation Analysis. In: Proceedings of the SPIE Vol. 7864A, Electronics Imaging Science and Technology Conference (IS&T), 3D Imaging Metrology. San Francisco, California.

Becker, S., 2009. Generation and application of rules for quality dependent façade reconstruction. Journal of Photogrammetry and Remote Sensing 64 (6), 640–653.

Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C., 2000. Image inpainting. Proceedings of SIGGRAPH, 417–424.
URL http://portal.acm.org/citation.cfm?doid=344779.344972

Bertsekas, D. P., Nedic, A., Ozdaglar, A. E., 2003. Convex Analysis and Optimization. Athena Scientific.

Böhm, J., 2008. Facade detail from incomplete range data. In: Proceedings of the ISPRS Congress, Beijing, China. Vol. 1. Beijing, China, p. 2.
URL http://www.isprs.org/proceedings/XXXVII/congress/5_pdf/115.pdf

Böhm, J., Becker, S., Haala, N., 2007. Model Refinement by Integrated Processing of Laser Scanning and Photogrammetry. In: Proceedings of 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-Arch). Zurich, Switzerland.

Bosché, F., Jan. 2010. Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. Advanced Engineering Informatics 24 (1), 107–118.
URL http://dx.doi.org/10.1016/j.aei.2009.08.006

Bosche, F., Haas, C. T., 2008. Automated retrieval of 3D CAD model objects in construction range images. Automation in Construction 17 (4), 499–512.
URL http://www.sciencedirect.com/science/article/B6V20-4PYYG3X-1/1/c9926c5f2bafbc0a941121af14

Brilakis, I., Lourakis, M., Sacks, R., Savarese, S., Christodoulou, S., Teizer, J., Makhmalbaf, A., Nov. 2010. Toward automated generation of parametric BIMs based on hybrid video and laser scanning data. Advanced Engineering Informatics 24 (4), 456–465.
URL http://dx.doi.org/10.1016/j.aei.2010.06.006

Budroni, A., Boehm, J., 2010. Automated 3D reconstruction of interiors from point clouds. International Journal of Architectural Computing 08 (01), 55–74.

Budroni, A., Böhm, J., 2010. Automatic 3d Modelling Of Indoor Manhattan-world Scenes From Laser Data. In: ISPRS Commission V Mid-Term Symposium: Close Range Image Measurement Techniques. Vol. XXXVIII. pp. 115–120.

Cantzler, H., 2003. Improving architectural 3d reconstruction by constrained modelling.

Cohen, W. W., 2005. Stacked sequential learning. In: International Joint Conference on Artificial Intelligence. pp. 671–676.

Davis, J., Marschner, S. R., Garr, M., Levoy, M., 2002. Filling holes in complex surfaces using volumetric diffusion. In: First International Symposium on 3D Data Processing, Visualization, and Transmission.

Debevec, P., Taylor, C. J., Malik, J., 1996. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: Proceedings of ACM SIGGRAPH. pp. 11–20.

DellAcqua, F., Fisher, R., 2002. Reconstruction of planar surfaces behind occlusions in range images. Transactions on Pattern Analysis and Machine Intelligence (PAMI) 24 (4), 569–575.

El-Hakim, S. F., Boulanger, P., Blais, F., Beraldin, J.-A., 1997. A system for indoor 3D mapping and virtual environments. In: Proceedings of Videometrics V (SPIE v. 3174). pp. 21–35.

Frueh, C., Jain, S., Zakhor, A., 2005. Data Processing Algorithms for Generating Textured 3D Building Facade Meshes from Laser Scans and Camera Images. International Journal of Computer Vision (IJCV) 61 (2), 159–184.

Furukawa, Y., Curless, B., Seitz, S. M., Szeliski, R., 2009a. Manhattan-world stereo. In: Proceedings of Computer Vision and Pattern Recognition. IEEE.
URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2009.html#FurukawaCSS09

Furukawa, Y., Curless, B., Seitz, S. M., Szeliski, R., 2009b. Reconstructing building interiors from images. In: Proceedings of the International Conference on Computer Vision (ICCV).
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459145

Furukawa, Y., Curless, B., Seitz, S. M., Szeliski, R., 2010. Towards Internet-scale Multi-view Stereo. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).

Golparvar-Fard, M., Pena-Mora, F., Arboleda, C. a., Lee, S., 2009. Visualization of Construction Progress Monitoring with 4D Simulation Model Overlaid on Time-Lapsed Photographs. Journal of Computing in Civil Engineering 23 (6), 391.
URL http://link.aip.org/link/JCCEE5/v23/i6/p391/s1&Agg=doi

Golub, G. H., van Loan, C. F., 1980. An analysis of the total least squares problem. SIAM Journal on Numerical Analysis 17, 883–893.

GSA, 2009a. GSA BIM Guide For 3D Imaging, version 1.0.
URL http://www.gsa.gov/bim

GSA, 2009b. GSA BIM Guide For Energy Performance, version 1.0.
URL http://www.gsa.gov/bim

Hähnel, D., Burgard, W., Thrun, S., 2003. Learning compact 3D models of indoor and outdoor environments with a mobile robot. Robotics and Autonomous Systems 44 (1), 15–27.

Huber, D., Hebert, M., 2003. Fully Automatic Registration of Multiple 3D Data Sets. Image and Vision Computing (IVC) 21 (7), 637–650.

Koppula, H. S. H., Anand, A., Joachims, T., Saxena, A., 2011. Semantic Labeling of 3D Point Clouds for Indoor Scenes. In: Neural Information Processing Systems.
URL http://pr.cs.cornell.edu/sceneunderstanding/nips_2011.pdf

Kou, Z., 2007. Stacked graphical models for efficient inference in markov random fields. In: In Proceedings of the 2007 SIAM International Conference on Data Mining.

Koutsourakis, P., Simon, L., Teboul, O., Tziritas, G., Paragios, N., Sep. 2009. Single view reconstruction using shape grammars for urban environments. In: International

Conference on Computer Vision (ICCV). Ieee, Kyoto, Japan, pp. 1795–1802.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459400

Moravec, H., 1996. Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids.

Munoz, D., Bagnell, J. A., Hebert, M., 2010. Stacked hierarchical labeling. In: European Conference on Computer Vision (ECCV).

Murphy, K., Torralba, A., Eaton, D., Freeman, W. T., 2005. Object detection and localization using local and global features. Lecture Notes in Computer Science (unrefereed). Sicily workshop on object recognition.

Nüchter, A., Hertzberg, J., 2008. Towards semantic maps for mobile robots. Robotics and Autonomous Systems 56, 915–926.

Nüchter, A., Surmann, H., Lingemann, K., Hertzberg, J., 2003. Semantic scene analysis of scanned 3d indoor environments. In: Proceedings of the Eighth International Fall Workshop on Vision, Modeling, and Visualization (VMV).

Pollefeys, M., Nister, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., Towles, H., 2008. Detailed Real-Time Urban 3{D} Reconstruction from Video. International Journal of Computer Vision 78 (2-3), 143–167.

Pu, S., Vosselman, G., 2006. Automatic extraction of building features from terrestrial laserscanning. In: IEVM06.

Pu, S., Vosselman, G., 2009. Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing 64 (6), 575–584.
URL http://www.sciencedirect.com/science/article/B6VF4-4W80C7C-1/2/c3090806c210aad2366e5179b9

Rabbani, T., , van den Heuvel, F. A., Vosselman, G., 2006. Segmentation of point clouds using smoothness constraint. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (5), 248–253.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S., 2007. Objects in context. In: Proc. ICCV.

Ripperda, N., Brenner, C., 2009. Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. In: Proceedings of AGILE Conference on Geographic Information Science. Hannover, Germany.

Roth, S., Black, M. J., 2005. Fields of experts: A framework for learning image priors. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). San Diego, California, pp. 860–867.

Rusu, R. B., Martona, Z. C., Blodowa, N., Dolhaa, M., Beetza, M., 2008. Towards 3D Point cloud based object maps for household environments. Robotics and Autonomous Systems 56 (11), 927–941.

Salamanca, S., Merchan, P., Perez, E., Adan, A., Cerrada, C., 2008. Filling holes in 3D meshes using image restoration algorithms. In: Proceedings of the Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT). Atlanta, GA.

Sappa, A. D., 2002. Improving Segmentation Results by Studying Surface Continuity. In: Proceedings of the International Conference on Pattern Recognition (ICPR). Vol. 2. pp. 929–932.

Snavely, N., Seitz, S. M., Szeliski, R., Jul. 2006. Photo tourism: exploring photo collections in 3D. Proceedings of SIGGRAPH 25 (3), 835–846.
URL http://portal.acm.org/citation.cfm?id=1141964
http://www.mendeley.com/research/photo-tourism-exploring-photo-collections-in-3d/

Stamos, I., Gene, Y., Wolberg, G., Zokai, S., 2006. 3D Modeling Using Planar Segments and Mesh Elements. In: Proceedings of 3D Data Processing, Visualization, and Transmission (3DPVT). pp. 599–606.

Tang, P., Huber, D., Akinci, B., 2007. A Comparative Analysis of Depth-Discontinuity and Mixed-Pixel Detection Algorithms. In: Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM). pp. 29–38.

Tang, P., Huber, D., Akinci, B., Lipman, R., Lytle, A., 2010. Automatic Reconstruction of As-built Building Information Models from Laser-Scanned Point Clouds: A Review of Related Techniques. Automation in Construction 19 (7), 829–843.

Thrun, S., Martin, C., Liu, Y., Hähnel, D., Emery-Montemerlo, R., Chakrabarti, D., Burgard, W., 2004. A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. IEEE Transactions on Robotics 20 (3), 433–443.

Trimble, 2011. Trimble Indoor Mobile Mapping Solution.
URL http://www.trimble.com/Indoor-Mobile-Mapping-Solution/Indoor-Mapping.aspx

Valero, E., Oliver, A. A., Huber, D., Cerrada, C., 2011. Detection, Modeling, and Classification of Moldings for Automated Reverse Engineering of Buildings from 3D Data. In: International Symposium on Automation and Robotics in Construction (ISARC).

Wolpert, D. H., 1992. Stacked generalization. Neural Networks 5, 241–259.

Xiong, X., Huber, D., 2010. Using context to create semantic 3d models of indoor environments. In: Proceedings of the British Machine Vision Conference. BMVA Press, pp. 45.1–45.11, doi:10.5244/C.24.45.

Xiong, X., Munoz, D., Bagnell, J. A., Hebert, M., 2011. 3-d scene analysis via sequenced predictions over points and regions. In: IEEE International Conference on Robotics and Automation (ICRA).

Yue, K., Huber, D., Akinci, B., Krishnamurti, R., 2006. The ASDMCon project: The challenge of detecting defects on construction sites. In: Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT).

Zhu, Z., Brilakis, I., Jan. 2010. Concrete Column Recognition in Images and Videos. Journal of Computing in Civil Engineering 24 (6), 478.
URL http://link.aip.org/link/?JCCEE5/24/478/1