Automatic Identification of Window Regions on Indoor Point Clouds Using LiDAR and Cameras

Richard Zhang
UC Berkeley
Berkeley, CA
rich.zhang@eecs.berkeley.edu

Avideh Zakhor
UC Berkeley
Berkeley, CA
avz@eecs.berkeley.edu

Abstract

In this paper, we propose an algorithm to automatically identify window regions on exterior facing facades of buildings using interior 3D point cloud resulting from an ambulatory backpack sensor system, outfitted with multiple LiDAR sensors and cameras. We develop a set of discriminative features for the task, namely visual brightness, infrared opaqueness, and an occlusion indicator, within a Markov Random Field (MRF) framework to provide structured prediction for window or glass regions. A preprocessing classifier is trained on the features to produce node potentials, and large margin parameter training is used to boost performance. Our algorithm has been trained on data taken at the 3rd floor of Cory Hall at UC Berkeley, with a total façade area of 269.1 m^2, and has been tested on walls taken on the 2nd floor of Cory Hall, a Walgreens, and an office building in San Francisco, with a total exterior façade area of 454.6 m^2. Window regions are successfully identified with 85.5% F_1-score and 94.2% accuracy.

1. Introduction

Three-dimensional modeling of buildings is an important problem with many applications in architecture, engineering and construction (AEC) industry. To model and simulate building energy efficiency with tools such as EnergyPlus, we not only need to construct a complete 3D model of the building, but also to semantically identify windows, lights and plug loads [5]. In particular, "window to wall ratio" of exterior facades is an important factor in energy consumption of most commercial buildings.

In this paper, we use 3D point clouds generated by an ambulatory human operated backpack mounted system made of multi-modal sensors, in order to detect windows on exterior facades of buildings [4][15]. A challenge in detecting windows from 3D colored point clouds is the limited amount of available training data. Furthermore, windows are innately shapeless and texture-less. The difference between "stuff" and "thing" categories in computer vision has been well documented [6]. "Things" are objects which have sizes and shapes, whereas "stuff" corresponds to materials with a homogenous or repetitive pattern, but no specific extent or shape. Windows correspond more to a "stuff" category than a "thing" category. As a result, rich high-dimensional feature descriptors for identifying objects such as those described in [3] do not apply to this task. In addition, windows do not have their own texture, as they are transparent. Similarly, in 3D space, windows do not have any unique features, so shape descriptors which have been developed for 3D point clouds [9][11] are not applicable.

To overcome these difficulties, we take advantage of the multiple modalities on our data acquisition unit, and develop a set of features which fuse LiDAR and camera data to identify window regions. To begin with, since windows are assumed to be unobstructed during acquisition, we use grayscale intensity values from the visual imaging modality to exploit lighting differences between indoors and outdoors. Secondly, the laser-beams from the LiDAR scanners incident on wall regions tend to receive a return, whereas beams incident upon glass windows tend to penetrate the surface. As such, we can use the percentage of laser beam returns received from each region on the wall as a strong feature, which we call infrared (IR) opaqueness. Thirdly, our last feature serves as an occlusion proxy, exploiting the fact that occluded regions are more likely to be wall than window.

We take a bottom-up segmentation approach with Markov Random Field (MRF), a commonly used graph-based formulation for structured prediction on point clouds. We use a Random Forest (RF) preprocessing classifier to produce node potentials. Inference corresponds to taking the maximum a posteriori (MAP) of the probability distribution, which can be solved efficiently and exactly using graph cuts [7]. We learn the parameter weights using a large margin learning algorithm [23], and use a constant pairwise term to enforce spatial consistency.

This paper is organized as follows. Section 2 describes related work in 3D point cloud classification. Section 3 describes the MRF framework, the way node and edge potentials are calculated, parameter training methodology, and feature extraction. Section 4 includes our experimental results. Finally, in Section 5, we conclude and discuss future work.

2. Related Work

A number of papers on recognition focus on designing high-dimensional features, such as HOG [3] for images and spin images for 3D point clouds [11]. However, these features do not describe windows well, as they are innately texture-less in 2D and shapeless in 3D.

There is a large body of work in scene understanding in various settings. Golovinskiy *et al.* use a pipelined approach to identify small outdoors objects, such as cars, mailboxes, and recycling bins; objects are first segmented from the background, features are extracted, and classifiers are trained [6]. Hierarchical classifiers are explored by Xiong *et al.* to provide sequenced predictions on images [27], and are then extended to perform on 3D point clouds outdoors [28]. Munoz *et al.* extend the algorithm on outdoor point clouds and images, which are acquired using a push-broom LiDAR scanner and camera [17].

In their seminal work, Anguelov *et al.* use pairwise associative MRFs to perform segmentation and classification tasks [2]. Munoz *et al.* add directional edge features, add higher order cliques for faster inference [19], and adapt functional gradient training [18]. Shapavalov *et al.* use correlation between object classes [21] and employ a Radial Basis Function (RBF) kernel to allow for nonlinear feature separability [22]. These methods are applied to outdoor data acquired from an aerial scanning platform. Lu and Rasmussen explore adding an RF or a Support Vector Machine (SVM) pre-processing classifier to calculate node potentials, rather than injecting features directly into the node potentials [16]. These methods focus on identifying objects such as vegetation, buildings, and vehicles, and cannot be directly applied to detecting windows in indoor scenes.

MRFs have also been used in a variety of indoor applications. Triebel *et al.* build associative MRFs to detect objects such as chairs, tables, and fans [24]. The method was also extended to the building space to provide semantic labels for the function of areas of buildings, such as lobby, corridor, and room. Previous work focused on identifying windows use a single modality in an outdoor setting, whereas we fuse data from multiple modalities in an indoor setting [1][14][26]. Wang, et al. [26] use LiDAR exclusively, and Lee and Nevatia [14] and Ali, *et al*. [26] use images to detect windows across multiple floors. The methods in [1] and [14] explicitly rely on the regular grid patterns of windows, an assumption which not only does not hold for many buildings, but also cannot be exploited from a single floor. The method in [26] uses a detector-based approach which works for outdoor environments, where windows appear to be small, localized objects.

Rusu *et al.* propose an algorithm which attaches semantic information to a point cloud for objects which are of interest to a robotic assistant, such as cupboards, tables, drawers, and shelves [20]. Top-down object detectors are fused with the bottom-up semantic segmentation framework by Lai *et al.* to detect objects such as bowls, caps, and coffee mugs on a desk using RGB-D video obtained from a Kinect sensor [13]. Koppula *et al.* also use MRFs in conjunction with a Kinect sensor to identify objects such as parts of chairs, tables, and computers [12]. The Kinect sensor provides denser depth information than the laser scanners on our backpack data acquisition unit, but does not have adequate sensing range to construct 3D models from walkthroughs. We choose to use the backpack acquisition platform, as it is capable of generating large scale, detailed, 3D models of building interiors rapidly.

3. Algorithm Description

A high level diagram showing the steps used for training and testing the proposed algorithm is shown in Figure 1. We use planar regions extracted from the 3D point cloud to represent the walls in space, as described in Section 3.1. The ground truthing procedure is described in Section 3.2. Section 3.3 discusses the MRF framework, along with inference and parameter training. Section 3.4 describes the features extracted from multiple sensing modalities which are used by our proposed algorithm.

3.1. Wall Decomposition

We start with a backpack, equipped with 2D laser range finders, IMU and cameras, which an ambulatory human operator carries through indoor environments, acquiring the raw sensor data, which is then automatically processed offline to generate 3D colorized point clouds [4][15]. The goal of the wall decomposition stage is to identify walls of the building, preferably facing the exterior, from the full 3D point cloud. An example of a room with an exterior facing façade is shown in Figure 2, which is a student lounge in Cory Hall, the Electrical Engineering building at

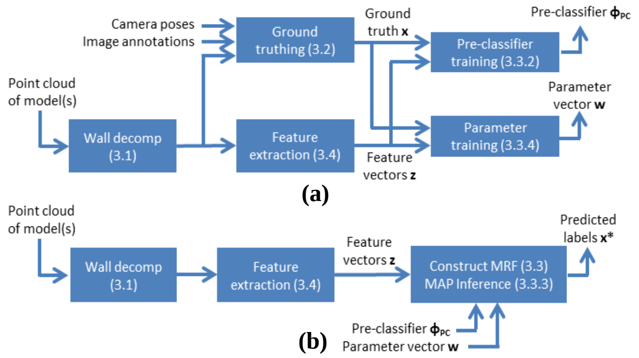


Figure 1 Top-level diagram **(a)** training **(b)** testing.

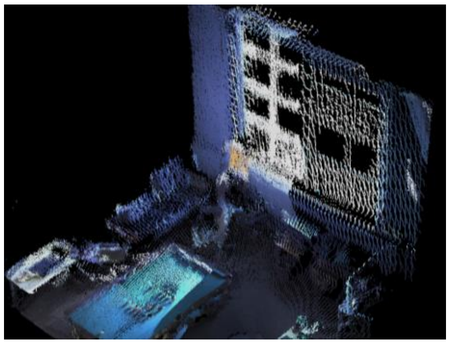


Figure 2 3D point cloud of a room from Cory Hall.

UC Berkeley. Each wall k within K total walls is described as a collection of $P_k \times Q_k$ pixels, each of which is an $r \times r$ centimeter 2D patch on wall k. Our goal is to automatically classify each "pixel" as a "window" or "wall". Wall decomposition can be performed either manually or automatically, as described below.

3.1.1 Manual Wall Decomposition

The point cloud of the model can be cropped into separate walls with commercial tools such as QuickTerrain Modeler[1], using approximately 1 minute of user time per wall. Parameters which describe each wall as a 2D planar region are then automatically extracted. RANSAC plane fitting is first performed, with tolerance of 0.6 meters [10]. The extent of the planar region is automatically determined by projecting the associated point cloud points onto the plane, with the outermost points defining the boundaries of the planar region. The planar region is then automatically divided into pixels of resolution r meters. Given a model with K walls, the dimensions of the k^{th} wall are denoted as H_k and W_k, for k in $\{1,2,...,K\}$. For wall k, the representative "pixelized" wall has dimensions $P_k \times Q_k$, where P_k = floor[H_k/r] and Q_k = floor[W_k/r], $R_k = P_k \times Q_k$ total pixels, and ordered pair (p,q) indexes a pixel. In our experimental dataset, the width of these walls range anywhere from 3.3 to 13 meters, and an average of 6.7 meters.

3.1.2 Automatic Wall Decomposition

Automatic 2.5D and 3D surface reconstruction algorithms have been developed to automatically extract the watertight building floor plan and vertical walls, given the 3D point cloud resulting from our backpack acquisition system [25]. Walls which lie on the outer perimeter can be automatically identified as follows: A single wall is first found by constructing a ray from the center of the model outwards in any direction. The outermost wall the ray intersects with is guaranteed to be on the outer perimeter, as are its adjacent walls. We trace along the adjacent walls, in either direction, until we return to the first wall. The traversed walls comprise the outer perimeter of the scanned area. From there, the planar region representation for each wall can be automatically extracted following the process described in Section 3.1.1. An example of manually and automatically extracted walls is included in Section 4.4.

[1] trademark by Applied Imagery LLC

3.2. Ground truthing

Ground truthing is the process of labeling each pixel *(p,q)* on each wall k as a "window" or a "wall". This is performed for training and performance evaluation and not testing. As part of the 3D point cloud generation process, the pose of the backpack is recovered as a function of time as it traverses the environment [4]. Since the cameras are rigidly mounted on the backpack, the pose for the collected imagery is known as well. For each wall k, we select one or two images with the wall in the field of view. We then annotate windows as polygons on the image, which are then projected onto the planar region. Pixels which are inside and outside of the polygons are ground truthed as "window" and "wall", respectively.

The associated camera pose derived from backpack localization algorithms can be error prone and thus lead to large projection errors [15]. As such, we refine camera pose values for ground truthing by manually finding correspondence points between the camera images and the 3D point cloud model, and solving a perspective-n-point (PNP) problem [8]. The pixelized ground truthed labeling of the example room in Figure 2 is shown in Figure 3(d).

3.3. Markov Random Field

3.3.1 General framework

MRFs are a common method of performing structured inference. Given a set of features **z**, the probability distribution of possible labeling configurations **x,** along with an equivalent energy function representation, is computed from potential functions through Equations (1) and (2) as follows:

$$P(\mathbf{x} \mid \mathbf{z},\mathbf{w}) = \frac{1}{Z(\mathbf{z},\mathbf{w})} \exp\left(-E(\mathbf{x},\mathbf{z};\mathbf{w})\right) \tag{1}$$

$$E(\mathbf{x},\mathbf{z};\mathbf{w}) = \sum_{i \in V} \psi_i\left(x_i, \mathbf{z_i}; \mathbf{w}\right) + \sum_{(i,j) \in E} \psi_{ij}\left(x_i, x_j, \mathbf{z_i}, \mathbf{z_j}; \mathbf{w}\right) \tag{2}$$

Sets V and E represent the set of all nodes and vertices, respectively in the graph, and ψ_i and ψ_{ij} are node and edge potential functions, respectively. Each pixel on each wall is a node on the graph, and is indexed as i in $\{1,2,\ldots, \sum_{k=1}^{K} R_k\}$. An edge is formed between neighboring pixels, in a 4-connected grid. Feature matrix **z** has dimensions $F \times \sum_{k=1}^{K} R_k$, where F is the number of features extracted. Feature vector $\mathbf{z_i}$ has length F, and is the feature extracted from the i^{th} pixel. Labeling for our problem of window detection is binary, $\mathbf{x} \in \{0,1\}^{\sum_{k=1}^{K} R_k}$, where 0 and 1 represent labelings of "wall" and "window", respectively. Indicator functions x_i^0 and x_i^1 are defined to be 1 if pixel i is labeled 0 or 1, respectively, and 0 otherwise. A common assignment of potential functions is a log-linear model.

$$\psi_i(x_i,\mathbf{z_i};\mathbf{w}) = \langle w_{node}^0, \mathbf{z_i}\rangle x_i^0 + \langle w_{node}^1, \mathbf{z_i}\rangle x_i^1 \tag{3}$$

An associative model is used for the *energy* term to penalize neighboring pixels with differing labels.

$$\psi_{ij}(x_i, x_j, \mathbf{z_i}, \mathbf{z_j}; w) = w_{edge} 1_{\{x_i \neq x_j\}} \tag{4}$$

Variables $w_{node}^0, w_{node}^1, w_{edge}$ comprise vector **w**.

3.3.2 *Preprocessing classifier*

We train a RF preprocessing classifier to obtain the *data* term $\psi_i(x_i,\mathbf{z_i};\mathbf{w})$. The pre-classifier reduces dimensionality of the input features, and enables incorporation of non-linear classification ability. Lu and Rasmussen have reported RF to have better performance than SVM pre-classifiers for 3D point clouds [16]. We use the *distribution* of votes resulting from the pre-classifier, $\Phi_{PC}(x_i|\mathbf{z_i})$, rather than the binary classification to compute the *data* term in the original MRF problem, as follows:

$$\psi_i(x_i,\mathbf{z};\mathbf{w}) = w_{node}^0 \Phi_{PC}(x_i = 1 | \mathbf{z_i}) x_i^0 + w_{node}^1 \Phi_{PC}(x_i = 0 | \mathbf{z_i}) x_i^1 \tag{5}$$

3.3.3 *Inference*

Inference is performed by finding the labeling configuration **x*** which maximizes the probability function, or equivalently, minimizes energy. This is referred to as the maximum a posteriori (MAP) labeling.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x} | \mathbf{z}, \mathbf{w}) = \arg\min_{\mathbf{x}} E(\mathbf{x}, \mathbf{z}; \mathbf{w}) \tag{6}$$

For a binary label problem, the MAP inference can be reformulated as a graph cut problem and solved efficiently and exactly [7].

3.3.4 *Parameter training*

We use the Margin Rescaled algorithm from Szummer, *et al.* [23] to train parameter vector **w**. The algorithm requires computation of the energy function (2) for any possible labeling **x**, which requires computation of the *data* term (5), which in turn requires evaluation of pre-classifier $\Phi_{PC}(x_i|\mathbf{z_i})$ for every pixel i in the training set. We use a k-folds procedure: with K walls in the training set, K pre-classifiers $\mathbf{\Phi_{PC}}$ are trained, leaving one wall out at a time. For each pixel i, the *data* term is then calculated using Equation (5) with pre-classifier Φ_{PC_k}, where pixel i originates from wall k. The Margin Rescaled algorithm uses an iterative, large margin learning procedure to find the optimal parameter vector **w.**

3.4. Feature Extraction

We use a total of F=3 features for each pixel i to build feature vector $\mathbf{z_i}$.

3.4.1 *Visual brightness*

The brightness level indoors differs from the lighting outdoors and can be used as a discriminative feature to identify windows. We project points within 0.6 meters of the wall onto the planar representation of the wall. For each pixel, the median grayscale value of the projected points found within 2 centimetres from the pixel centre, in Manhattan distance, is used as the feature value. Due to the limited amount of training data, other colour components are not used. This is done to prevent the classifier from over-fitting on the colour of the walls within the training set. With additional training data, other colour features can be used, and the classifier can exploit the colour difference between indoor walls and outdoor scenes.

3.4.2 *Infrared opaqueness*

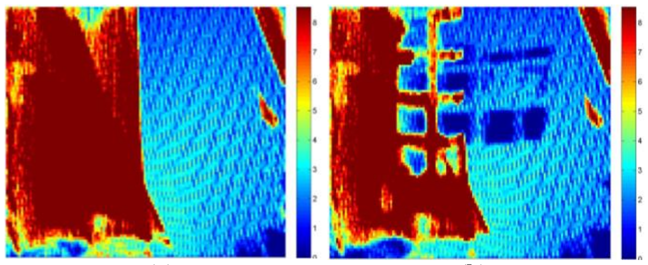
Laser-beams from the LiDAR sensor typically penetrate glass, whereas a return is frequently received from a "wall" region. For each wall k, the number of returned laser-beams and number of incident laser-beams on each pixel are represented by matrices $\mathbf{N_k}$ and $\mathbf{M}_k$, respectively, both of which are dimensions $P_k \times Q_k$, as defined in Section 3.1.1. Matrix $\mathbf{N_k}$ is obtained by projecting the points in the point cloud which are within 0.6 meters of the planar region representation, and counting the number of points which fall within the bounds of each pixel. Matrix $\mathbf{M_k}$ is obtained by projecting the position and direction of the laser scanner, at each laser scan time, onto the planar region, and finding the intersecting pixel (p,q). The value of $\mathbf{M_k}$ is then incremented at (p,q).

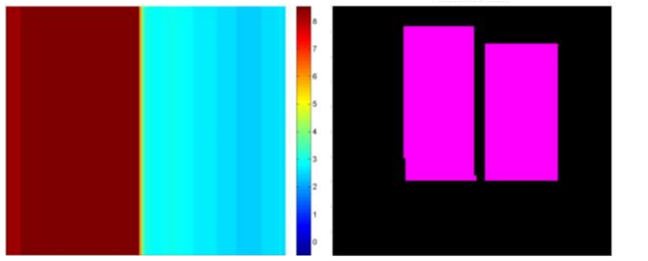
For scenarios where the backpack operator is far from the wall of interest, successive laser-beams may be spaced far apart enough such that many intermediate pixels have no associated incident laser-beams. A similar effect would be observed for small pixel resolution r, since each laser-beam can only be associated to a single pixel. To mitigate this, we convolve matrices $\mathbf{N_k}$ and $\mathbf{M_k}$ with a Gaussian kernel with σ=2.5 centimetres to produce matrices $\mathbf{N_k}'$ and $\mathbf{M_k}'$, so that feature is invariant to resolution size r. Examples for the wall pictured in Figure 2 is shown in Figure 3(a) and 3(b), respectively. The opaqueness feature is obtained by an element-wise division of the matrices. Specifically, the opaqueness value for pixel (p,q) on wall k is $\mathbf{N_k}'(p,q)/\mathbf{M_k}'(p,q)$. The opaqueness map for the example room is shown in Figure 3(e). Many of the window regions are IR transparent, as expected. However, at times, the LiDAR may still receive returns from "window" pixels. For example, if the window is tilted open, an incident laser is more likely to result in a return.

3.4.3 *Occlusion proxy*

The final feature acts as a proxy to occlusion. Our proposed algorithm is designed to be able to process scans from indoor areas, which may be cluttered by objects such as desks, chairs, people, and plants. As a result, collected features, particularly the opaqueness data, may be errant. For pixel (p,q) on wall k, the opaqueness feature divides $\mathbf{N_k}'(p,q)$ laser returns by $\mathbf{M_k}'(p,q)$ incident laser-beams. For uncluttered areas, a pixel may receive hundreds of incident beams, and as a result, the derived opaqueness feature is an accurate reflection of its properties. However, for occluded areas, $\mathbf{M_k}'(p,q)$ may be extremely low, in which case the opaqueness feature becomes overly sensitive to whether individual laser beams received a return or not, and may not be an accurate indication of the material properties. The classifier downstream is then unable to differentiate between 0 returns from many, e.g. 100, and 0 returns from very few, e.g. 2 incident laser-beams. In the former case, the opaqueness feature would correctly indicate that the pixel corresponds to a "window" region, whereas in the latter case, there is a strong possibility that the pixel is simply a "wall" region, occluded by some other object.

To overcome this, we compute occlusion proxy through the following procedure. For a given wall k, the number of laser beam counts, $\mathbf{M_k}'$, is obtained from the opaqueness feature; an example of this is shown in Figure 3(a) for a wall of height H_k meters and P_k pixels in the vertical direction. In this particular example, due to the operator trajectory, the number of scanned points is drastically different on the left and right halves of the wall. As such, dividing the entire laser beam count matrix $\mathbf{M_k}'$ by a single number would not successfully elucidate which portions of the wall are occluded. Rather, a normalizing matrix $\mathbf{L_k}$ is first computed. The median value of laser beam counts is obtained along a tall and narrow sliding window of size H_k×0.25 meters, slid along the x-direction. This value is assigned to all P_k values in the vertical direction in matrix $\mathbf{L_k}$, as shown in Figure 3(c). The occlusion feature for a pixel (p,q) on wall k, designated as $\mathbf{O_k}$, is then set to $\min(1,\mathbf{M_k}'(p,q)/(\mathbf{L_k}(p,q)+\varepsilon))$. The function highlights regions which have a low number of laser-beam counts. Variable ε is set to 10^{-3} to avoid divide by 0. As seen in Figure 3(f), regions with low values are on the bottom of the image, and coincide with the presence of occluders.


(a) (b)


(c) (d)

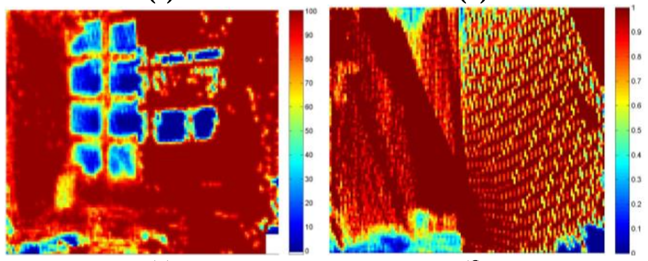

(e) (f)

Figure 3 (a) Number of incident laser-beams; **(b)** number of laser-beam returns; **(c)** number of incident laser-beams, filtered; **(d)** ground truth (pink – windows, black – wall); **(e)** IR opaqueness feature map; **(f)** occlusion proxy feature.

4. Experimental results

The data for training set for all classifiers has been taken from the 3rd floor of Cory Hall. The walls of Cory comprised of 1.67 M points, and were segmented from 44 M points in the original model. The test set data has been taken from the 2nd floor of Cory Hall, a Walgreens, and an office. Statistics of the datasets, using manual decomposition, as described in Section 3.1.1, are shown in Table I.

	Training	**Testing**			
set name	**3rd Floor**	**2nd Floor**	**Wal-greens**	**Office**	**Total**
# walls	**10**	9	3	5	**17**
Total area [m²]	**269.1**	316.3	80.1	58.2	**454.6**
Window area [m²]	**65.0**	67.4	11.6	16.9	**96.0**
% window	**24.1%**	21.32%	14.53%	29.09%	**21.12%**

Table I Training and testing set statistics.

For testing, the manual decomposition is used for baseline results reported in Sections 4.1 and 4.2, the ablation study in Section 4.3, and additional test set in Section 4.5. Results using automatically decomposed walls are discussed in Section 4.4. Resolution r is set to 4 centimeters; as discussed in Section 3.4.2, feature extraction is invariant to resolution. Training on a 2.66 GHz Intel processor took 13.8 minutes in total: 1.25 minutes per classifier for 11 classifiers, 1 classifier for pre-classifier training and 10 more for the leave-one-out process for parameter training.

4.1. Random Forest pre-classifier

A total of 50 trees were trained in the RF pre-classifier. The training subset for each tree was 80% of the total training set. Trees are extended until each leaf node had a maximum of 10 examples. Results of the RF pre-classifier are shown in Table II. Since we use three features for the classification, the output from the RF classifier can be directly plotted. Figure 4 displays a heat map for $\Phi_{PC}(x_i=1|\mathbf{z_i})$ given different slices of the value of the occlusion proxy feature. Note that *lower* values of occlusion proxy indicate that the pixel is more occluded. In Figure 4(a), the occlusion proxy is at its maximum value, indicating the region on the wall is un-occluded. Pixels which are IR transparent and visually bright are classified as very likely “window”, as seen in Figure 4(b). As the occlusion proxy feature decreases, the classifier assigns lower probability to “window”, for the same brightness and IR opaqueness values. Figure 4(c) shows that heavily occluded regions need to be completely transparent and bright to be classified as “window”.

4.2. Semantic segmentation results

Performance results for the pre-classifier round($\Phi_{PC}(x_i=1|\mathbf{z_i})$), the MRF with default weights, and the MRF with trained weights are shown in Table II. The first group of rows corresponds to the parameter vector weights. The second group of rows shows commonly used metrics for performance evaluation.[2]

The second column in Table II refers to results of the per-pixel RF pre-classifier only. This is equivalent to forming a MRF, but setting the edge weights to 0. In this case, the energy function only depends on the node potentials. The third column shows results for the MRF given default untuned weights. Improvement is seen in the F1-score and accuracy, as compared to RF only, indicating that, as expected, contextual information from neighboring classifications are useful. The last column in Table II shows the weights from the Margin Rescaled algorithm described in Section 3.3.4 [23]. The node weight w_{node}^0 is larger than w_{node}^1, increasingly penalizing a x_i=0 label choice and resulting in a slight bias of labeling regions “window”. F1-score and accuracy are slightly improved with the trained weights, as compared to default weights, indicating that parameter training has been successful. The results for the MRF with trained weights are broken down by test area in Table III.

The wall which contains the entrance of a Walgreens, along with its semantic segmentation, is shown in Figure 5(a), 5(b) and 5(c). The wall contains a window, a mirror, and glass doors into the store. Various posters and banners are taped onto the doorway, as shown in Figure 5(b). These are ground truthed as “wall”, since the algorithm is trained and designed to identify *unobstructed* windows. The pixels which correspond to the window on the left of

[2] TP – true positive, FN – false negative
FP – false positive, TN – true negative
Precision = TP / (TP + FP), Recall = TP / (TP + FN)
F1-score = TP / (TP + .5 FP + .5 FN)
Accuracy = (TP + TN) / (TP + FN + FP + TN)

	no pairwise terms / RF only	**untrained weights**	**trained weights**
w_{node}^0	1	1	.635
w_{node}^1	1	1	.537
w_{edge}	0	1	.555
Precision	87.0%	91.3%	90.2%
Recall	74.7%	79.3%	81.2%
F1-score	80.4%	84.9%	85.5%
Accuracy	92.3%	94.0%	94.2%

Table II MRF classification results.

Dataset	**2nd floor**	**Walgreens**	**Office**	**All**
TP [m^2]	54.5	8.8	14.7	78.0
FN [m^2]	13.0	2.9	2.2	18.0
FP [m^2]	3.9	1.8	2.8	8.5
TN [m^2]	245.0	66.6	38.4	350.1
Precision	93.4%	83.2%	83.8%	90.2%
Recall	80.8%	75.4%	87.1%	81.2%
F1-score	86.6%	79.1%	85.4%	85.5%
Accuracy	94.7%	94.2%	91.4%	94.2%

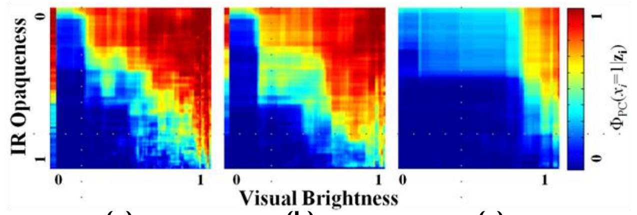
Table III Classification results per test data set.

Weights	**Statistic**	**br**	**op**	**br+op**	**all3**
No pairwise terms / RF only	**Precision**	64.5%	76.4%	82.4%	**87.0%**
	Recall	55.4%	69.0%	73.8%	**74.7%**
	F1-score	59.6%	72.5%	77.9%	**80.4%**
	Accuracy	84.1%	89.0%	91.1%	**92.3%**
Trained weights	**Precision**	67.3%	88.3%	**90.6%**	90.2%
	Recall	52.3%	75.8%	**81.2%**	**81.2%**
	F1-score	58.8%	81.6%	**85.7%**	**85.5%**
	Accuracy	84.6%	92.8%	**94.3%**	**94.2%**

Table IV Test results with limited feature sets: **br**-brightness, **op**-opaqueness, **br+op** – brightness & opaqueness, **all3** – brightness, opaqueness & occlusion proxy.

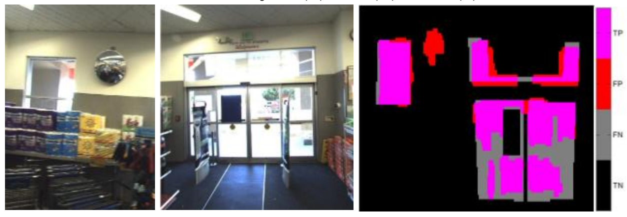
Decomp. Method	**Auto (out)**	**Inter (ext)**	**Manual(ext)**
% window	5.08%	22.64%	21.12%
Precision	75.5%	91.5%	90.2%
Recall	80.4%	80.4%	81.2%
F1-score	77.9%	85.6%	85.5%
Accuracy	97.7%	93.9%	94.2%

Table V Test results with different wall decomposition methods.



(a) (b) (c)

Figure 4 Output from the per-pixel RF pre-classifier (Red – $\Phi_{PC}(x_i=1|\mathbf{z_i})$ = 1: likely to be window, Blue – $\Phi_{PC}(x_i=1|\mathbf{z_i})$ = 0: likely to be non-window). Occlusion Proxy is **(a)** 1.0 **(b)** 0.74 **(c)** 0.47.



(a) (b) (c)

Figure 5 (a)(b) Visual images of example wall; **(c)** semantic segmentation of example wall in Walgreens.

the wall are successfully identified. The mirror, however, is incorrectly identified as a window, predominantly because a mirror is seen as IR transparent. Incident laser beams often reflect off the mirror; a return is received and computed to be behind the mirror, at a distance twice as far from the laser scanner. This is a common failure case.

4.3. Ablation Study

To evaluate the effectiveness of each feature, an ablation study is performed, where the algorithm is trained and tested on limited feature sets, as shown in Table IV. The study is performed on the manually decomposed walls described in Section 4.2. The results in the top group of rows in Table IV correspond to the per-pixel RF pre-classifier only, and the bottom corresponds to MAP inference with the MRF. The highest value, or any value within 0.2% the highest value, in each row is bolded. In the top group of rows, the highest F_1 and accuracy scores are achieved when all features are used, as shown in the last column. Interestingly, the MRF results in system performance with and without the occlusion proxy to be roughly equivalent. Opaqueness is stronger feature than brightness. Both are stronger than the occlusion proxy, which was designed to help refine the opaqueness feature.

4.4. Results with Automatic Wall Decomposition

Thus far, we have shown results for manually selected exterior facing walls as described in Section 3.1.1. An example of this for the second floor of Cory Hall is shown in cyan in Figure 6. Alternatively, we can automatically generate a complete set S of all vertical walls from the 3D point cloud by applying the method in [25]. In Figure 6, set S consists of thin black lines, thick black lines, and blue lines. Thick black lines are a subset of S, corresponding to exterior facing walls, assuming prior user knowledge of the building layout. These are identified with a simple interactive GUI tool, where the user can rapidly select exterior facing walls. Dark blue lines are also a subset of S, corresponding to the walls of the outer perimeter of scanned path, which are automatically identified with no user intervention, as described in Section 3.1.2. Thin black lines are a subset of S corresponding to interior facing walls of the scanned path.

A performance comparison on different decomposition methods is shown in Table V. As expected, the interactive and manual decomposition methods have similar performance when evaluated on the exterior facing walls, as shown in the last two columns of Table V. As shown in the second column of Table V, when testing on outer perimeter walls resulting from completely automatic decomposition, the opportunity for false alarms increases by 430%, precision drops from 91.5% to 75.5%, and F_1-score drops from 85.6% to 77.9%. Accuracy increases,

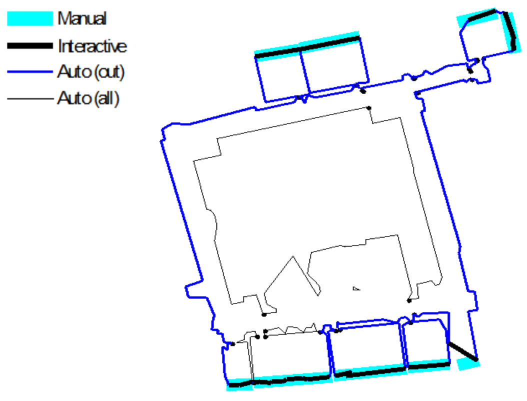


Figure 6 Manual vs. Automatic wall decomposition on 2nd floor of Cory Hall. Auto – automatic decomposition, out – outer perimeter walls, all – all walls.

because the additional walls are classified with 98.8% accuracy. In this case, results are achieved completely automatically, with no human intervention in manually cropping walls and no *a priori* information as to which walls face the exterior.

The semantic segmentation results using automatic wall decomposition for each pixel on the 2D walls in Figure 6 are transformed into their original position in 3D in Figure 8. There are several false alarms, as indicated by the red regions. Many of these correspond to indoor glass and can be removed by better detecting which walls can possibly correspond to the exterior of a building. In addition, since the false alarms almost all correspond to small regions, with prior knowledge of minimum window size, post-processing can be performed to increase precision.

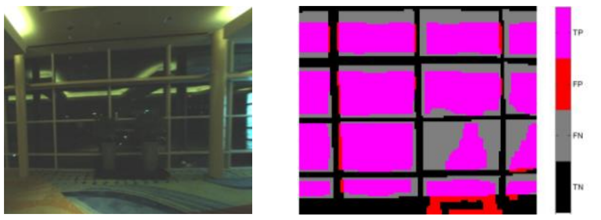
4.5. Results on Dataset Acquired at Night

The proposed algorithm uses the brightness variability between indoor and outdoor environment to identify windows. Because the training dataset was acquired during the day, the RF classifier learns that window regions correspond to where brightness level is higher, as seen in Figure 4. We now characterize performance of our proposed algorithm on an additional test set, which was acquired at a hotel in Houston during the night time. The set also has the distinction of being majority “window”, rather than majority “wall” as were the training and earlier test sets, shown in Table I and Table VI. Semantic segmentation results are shown in Table VI, and an example image from the data acquisition, along with its corresponding semantic segmentation, are shown in Figure 7(a) and 7(b), respectively. There are partial occluders, such as the potted plants, which cause false negatives. However, the classifier is able to distinguish between the object classes with some accuracy, due to the discriminative ability of the opaqueness feature. As seen in Table VI, the brightness feature reduces performance, and removing it increases F_1-score from 73.9% to 81.1%.

Gathering additional training data under different conditions is likely to improve performance. Also, during test time, given prior knowledge, it is possible to use a different classifier, depending on whether it was bright or dark outside.

% window	74.6%	
Feature set	**op+occl**	**all3**
Precision	84.4%	95.5%
Recall	78.0%	60.3%
F_1-score	81.1%	73.9%
Accuracy	72.8%	68.3%

Table VI Houston dataset semantic segmentation results, **op+occl** – opaqueness & occlusion proxy, **all3** – brightness, opaqueness & occlusion proxy.



(a) **(b)**

Figure 7 **(a)** Visual image **(b)** Segmentation of example wall.

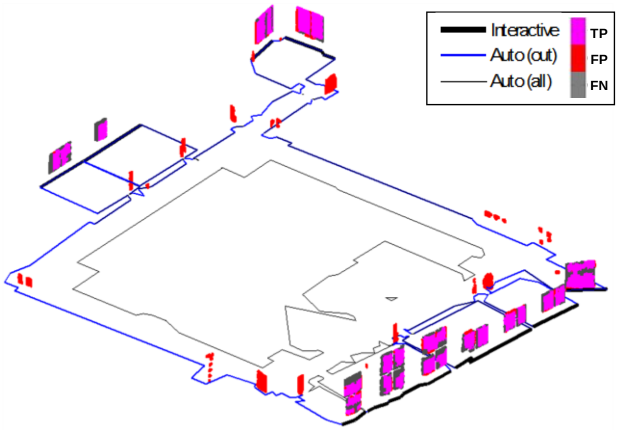


Figure 8 Semantic segmentation of walls in 2nd floor of Cory Hall using automatic wall decomposition.

5. Future Work

Future improvements are likely to be obtained by incorporating additional sensing modalities, using additional features, and training on additional datasets taken under varying conditions.

References

[1] H. Ali, C. Seifert, N. Jindal, L. Paletta, and G. Paar. Window Detection in Facades, In *ICIAP*, 2007.

[2] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3-d scan data. In *CVPR*, 2005.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[4] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor. Indoor localization algorithms for a human-operated backpack system. In *3D PVT*, 2010.

[5] EnergyPlus Energy Simulation Software. [online] 2013, http://apps1.eere.energy.gov/buildings/energyplus/ (Accessed 01 Sept 2013).

[6] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *ICCV*, 2009.

[7] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *J. R. Statistic Soc B*, 1989.

[8] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (DLS) Method for PnP. In *ICCV*, 2011.

[9] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, 2004.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Comm. of ACM*, 1981.

[11] A. Johnson and M. Hebert. Using spin images for efficient object recognition in clutter 3D scenes. In *IEEE PAMI*, volume 21, pages 433-449, 1999.

[12] H. S. Koppula, A. Anand, T. Joachims and A. Saxena. Contextually guided semantic labeling and search for 3D point clouds. In *IJRR*, 2012.

[13] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3D scenes. In *ICRA*, 2012.

[14] S. C. Lee and R. Nevatia. Extraction and Integration of Window in a 3D Building Model from Ground View Images, In *CVPR*, 2004.

[15] T. Liu, M. Carlberg, G. Chen, Jacky Chen, J. Kua, A. Zakhor. Indoor localization and visualization using a human-operated backpack system. In *IPIN*, 2010.

[16] Y. Lu and C. Rasmussen. Simplified markov random fields for efficient semantic labeling of 3D point clouds. In *IROS* 2012.

[17] D. Munoz, J. A. Bagnell, M. Hebert. Co-inference for multi-modal scene analysis. In *ECCV*, 2012.

[18] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009.

[19] D. Munoz, N. Vandapel, and M. Hebert. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *ICRA*, 2009.

[20] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, M. Beetz. Towards 3D Point cloud based object maps for household environments. In *Robotics and Autonomous Systems*, volume 56, pages 927-941. 2008.

[21] R. Shapovalov, A. Velizhev, and O. Barinova. Non-associative markov networks for 3d point cloud classification. In *ISPRS Comm III Symp PCV*, 2010.

[22] R. Shapovalov and A. Velizhev. Cutting-plane training of non-associative markov network for 3d point cloud segmentation. In *3D IMPVT*, 2011.

[23] M. Szummer, P. Kohli, and D. Hoiem. Learning large-margin random fields using graph cuts. In *ECCV*, 2008.

[24] R. Triebel, R. Schmidt, O. Martinez Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2d and 3d laser range data. In *IJCAI*, 2007.

[25] E. Turner and A. Zakhor. Floor plan generation and room labeling of indoor environments from laser range data. In *GRAPP*, 2014.

[26] R. Wang, J. Bach, and F. Ferrie. Window Detection from Mobile LiDAR Data. In *WACV*, 2011.

[27] X. Xiong and D. Huber. Using context to create semantic 3d models of indoor environments. In *BMVC*, 2010.

[28] X. Xiong, D. Munoz, J. A. Bagnell, and M. Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *ICRA*, 2011.