Detection of windows in point clouds of urban scenes

Agis Mesolongitis
Graduate Center of CUNY
New York, NY
amesolongitis@gc.cuny.edu

Ioannis Stamos
Graduate Center of CUNY / Hunter College
New York, NY
istamos@hunter.cuny.edu

Point Cloud Processing Workshop, CVPR 2012

Abstract

Laser range scanners have now the ability to acquire millions of 3D points of highly detailed and geometrically complex urban sites, opening new avenues of exploration in modeling urban environments. However, raw data are dense and complex, lacking high-level descriptive power, thus revealing the need for the automatic detection of architectural objects, such as facades, windows, balconies, etc. In this paper, we describe novel algorithms for the detection of windows, which are ubiquitous in urban areas. Detecting isolated windows is a challenging problem due to the inability of the laser range sensors to acquire any data on transparent surfaces and due to the wide variability of window features. Our approach is based on the assumption that the elements (windows) are arranged in multiple unknown periodic structures making our system robust to single window detection errors. This kind of detection is essential for high-level recognition algorithms, compression methods, registration, as well as realistic visualizations.

1. Introduction

The photorealistic reconstruction of individual buildings or large urban areas can be achieved by a variety of acquisition methods and interpretation techniques mainly based on ground-based and/or air-borne laser and image sensing. The state of the art in this area is surveyed in [7] and the ultimate goal is the reconstruction of detailed models of urban sites.

Our work focuses in data which were acquired by laser scanners and can be used for generating accurate 3D models. Since unprocessed models are heavy, complex and lack essential high-level descriptive power, our *goal* is to abstract the complex 3D models into high-level descriptors. This not only allows compressed representations but also facilitates higher level recognition processes. It also leads to more realistic 3D visualizations, which take advantage of the semantics of the scene.

In this paper, we are attacking one aspect of this important problem by detecting the locations of windows on building facades, assuming that the windows form *multiple two-dimensional periodic structures*. Exploiting these structures is very powerful, as it can reveal fully or partially occluded windows which might otherwise escape detection. It can also reduce the representation of a facade into a few descriptors and a base point cloud of the sample window. We also assume that (a) the laser scanner organizes the points in vertical scanlines (a 2D array), which facilitates our pre-processing steps and (b) the facades are mostly planar.

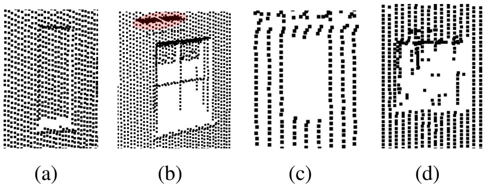


Figure 1. A selection of the various cases of windows in range scans: high resolution closed (a) and open (b) window with interior building points captured as well (shadowed in red - see pdf for color). Low resolution in (c) and (d).

3D laser range sensing results in complex representations full of clutter (cars, traffic lights, vegetation), missing data due to physical properties (for instance glass does not provide any measurements) and variability of appearances (e.g. open/closed windows — see Fig.1). Our problem is made significantly harder by also taking into consideration the missing data due to occlusions and the variability in data resolution (farther objects are scanned in lower resolution than close-by ones). However, the methods we describe in this paper address all these challenges. We demonstrate our approach using real data from a large urban scene.

2. System Overview

We developed an iterative algorithm that given the point cloud of a large urban scene, it automatically selects the points of each facade (see Fig.2) and outputs a set of 3D points that correspond to the estimated window centers. The main algorithmic module for the window center estimation is inspired by the Generalized Hough Transform [1]. Our system, initially pre-processes the facade points to ensure that all windows are uniformly represented as holes. Then, it converts the processed points to a 2D binary orthographic point occupancy map (Fig.6), it extracts and iteratively refines the 2D window centers using *local lattice fitting* and *lattice voting* (see Fig.7) and finally projects the 2D centers back to 3D.

The use of the 2D occupancy map enables the uniform representation of holes in point clouds of great resolution variance as watertight 2D regions. Consequently, this map allows us acquire an initial estimate of the window centers that needs to be refined and maybe augmented by recovered centers, undetected in the first pass. The refinement of the centers and the recovery of missed ones are done by a voting scheme in which each center votes for lattices that are considered to be good fits for the image. These lattices represent the local periodicity in the vertical and horizontal directions (see [18]). Therefore, our method takes into account the local window pattern. The paper is organized as follows: Section 3 summarizes related work, section 4 presents the automatic facade segmentation, section 5 deals with the 2D map generation and finally section 6 analyzes the iterative window center refinement.

3. Related Work

Detecting windows on facades is a problem that has received significant attention lately. Previous works have attempted to tackle the problem of detecting multiple regular windows structures in 2D images by using color-based features. For example, [8] and [16] suggest accurate methods based on the use of SIFT and other features and [17] uses multiple images jointly with the extracted SFM point clouds. [18] is using simple features such as Harris corners to detect multiple regularities but the steps of this algorithm implicitly rely on grouping based on the RGB values.

On the contrary, we rely solely on geometry and our algorithms are applied to a binary 2D map acquired from the 3D data. The windows are represented by connected regions of value 1 whose borders can be very crude in cases of low resolution (see Fig.10(a)). So, this automatically excludes clustering SIFT and other color based features as a first step to identify different groups of windows (as in [8]).

Furthermore, there has also been a big amount of work such as [6] and [5] where a single rectilinear window patterns are assumed. These approaches however could not work for multiple structures and their features still rely on RGB data. General approaches include [4], which detects windows as blobs in a color image but does not exploit the underlying structure. Shape grammar methods have also been examined, such as [14], which uses training images as well as specification of grammar rules and [11], which relies on manual interaction. However, in these approaches periodicity is not enforced.

Fewer approaches exist when the input is a laser range scan. [3] proposes an efficient approach for the detection of a single period in the vertical direction for each window column of a 3D facade. [9] derives regularities of substructures from a 3D model or range scan of a scene. This general approach can be used for extracting regularities but it is sensitive in the calculation of curvatures. Simpler approaches like [10] identify isolated windows directly in the 3D data by examining holes in a facade but do not detect regular groups and would not be robust in cases of occlusion. In [13] window-like rectangular features were extracted by using 3D edge detection on high-resolution 3D data. The features were used for 3D-to-2D registration and no regularity was enforced. In [15] a Markov Network approach that requires training is used to label points as windows. In [19], 3D repetitive elements are manually selected and automatically consolidated. Finally, in [12] facades are adaptively partitioned in by horizontal and vertical planes based on the boundary features of planar regions. However, the data used in [12] demonstrate a uniform acquisition resolution, which is not the case in our datasets. These methods can be seriously affected by the variation in the resolution and the window appearance, which can both exist inside a single scan.

Our *contributions* can be summarized as follows:
(a) we efficiently extract the facades of a large-scale point cloud, (b) we extract 3D windows in large-scale datasets, (c) we detect 3D windows in high-, low- and mixed-resolution data, (d) by modeling regularity, we are able to detect windows that are missing due to occlusions or low resolution, (e) we present a quantitative evaluation.

4. Pre-processing: Urban Scene Segmentation

We segment the points that correspond to each facade by applying the algorithms described in the following sections. These techniques can be applied in large data sets yielding results like the one in Fig.2(d). At this point, we have to note that the sensor: a) is calibrated so that the z-axis corresponds to gravity direction and b) stores the points in a 2D array, thus revealing connectivity.

4.1. Building Points Detection

A first step towards the segmentation of each single facade of an urban scene is to detect the points that belong to buildings. At this point we exploit the structure of the input data. Specifically, for each scan line, we check each point sequentially and update an average of the projections on the horizontal plane. Each point will be weighted depending on how close it is from the previous one along the scan line. The weights will be of the form: $w_k = e^{-\frac{\|\mathbf{p}_k - \mathbf{p}_{k-1}\|^2}{\sigma_w^2}}$ where

$\mathbf{p}_k = (x_k, y_k)$ is the projection of point k and σ_w is a parameter which in our experiments was set to be 0.3 times the expected thickness of the facade. When we reach the points of the buildings, these weights become very large (since all the facade points will have similar projection) and the average converges to the location of the facade. We then use a tolerance of 1 meter from this converged point and we label the points inside this range as facade points (see Fig.2(b)). This method is accurate and efficient for all resolutions of the 3D data.

4.2. Automatic Point Clustering

The next step is to operate on the facade points to acquire the orientation of each facade. Firstly, we fit planes in small neighborhoods around each point using Principal Components Analysis. If the plane fit is good (smallest eigenvalue of local covariance matrix is smaller than a threshold), then we assign a normal to the point. Finally, normals with an angle $\theta_z \in [60, 120]$ from the gravity axis are fed to a mean-shift mode seeking algorithm ([2]) and representative orientations (azimuth angles) are estimated (the modes are usually 1 or 2). Based on these representatives, we cluster all the points of the data set.

4.3. Region Growing Segmentation

As a final step, we apply a region growing segmentation algorithm to connect points which 1) have a small distance between them and 2) are assigned to the same orientation cluster. In order to get the final facade volumes, we find the oriented bounding boxes of each segment and merge overlapping boxes. The result is one box per facade which in practice contains all the facade points and maybe some outliers (see Fig.2). Having divided the point cloud into a number of facades, we need to detect the windows in each one of them. This exactly is the goal of the next sections.

5. Conversion to 2D / Initial Window Detection

As we desire to unify the representation of windows, we convert all of them into holes lying on a planar facade (windows as holes can also be seen in [11]). In this way, we will not distinguish between closed and open windows, missing or noisy window points. In order to accomplish that, it is essential that we accurately detect the major plane of the facade. This is done as follows: First, we find the strongly planar points using a process similar to the one used in Sec.4.2. The points are used to fit a single major plane. Having this plane, we can effectively classify all the points into these categories: *behind the plane*, *on the plane*, *in front of the plane*. We choose to delete all the *behind the plane* points since these are mainly the ones acquired through the window frames (see Fig.1).

The processed point cloud can now be used to identify the window regions. We create a 2D binary point occupancy map, where the window regions are represented by pixels with value 1. To achieve this, we firstly take an orthographic snapshot of the point cloud from the direction of the normal of the major plane. The pixel dimensions are 1×1 centimeters, which is an accurate resolution for urban scenes. In order to have watertight regions, instead of projecting points, we use the connectivity grid given by the laser scanner and for each three connected points we project the bounding rectangle and assign the value 0 to the corresponding pixels. All pixels not accessed during this projection module are assigned the value 1 and will correspond to“missing point”.

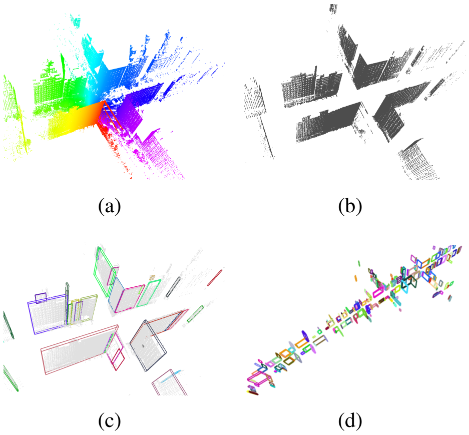


Figure 2. (a) Original point cloud with each scan line having a different color, (b) building points (Sec. 4.1), (c) segmented facades (Sec. 4.3), (d) The result of the algorithm applied to a large dataset

Having this binary map, we find connected regions of pixels valued with 1 and we assume that the ones touching the top of the image are considered to be *sky* points and are regions where windows should not exist. The rest of the 1-valued closed regions are considered to be the *window regions*, i.e. the initial estimates for the windows. We define *the image which has value* 0 *everywhere, except for the valid window regions* (which will have value = 1), as $I_w(x, y)$. The size of I_w will be $N \times M$ and examples of I_w can be seen in Figs.6 and 10(a). The centers of the bounding boxes of the window regions are used as the input for the consequent steps of our pipeline and are denoted by $\mathbb{C}_0 = \{(x_k, y_k)\}$, with $|\mathbb{C}_0| = K_{C_0}$, being the cardinality of the set $\mathbb{C}_0$. *The bounding boxes will be used only for the initialization step and then be discarded, as they are not always accurate due to low resolution variations and occlusions*. In order to reject tiny regions or other holes due to occlusions, we specify some boundaries for the width and the area of the bounding boxes and remove the non-complying centers from $\mathbb{C}_0$. An example of the elements of $\mathbb{C}_0$ can be seen in Fig.6.

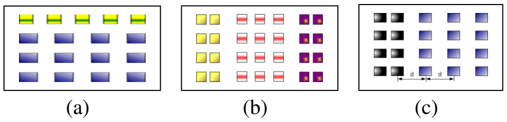


Figure 3. Examples of window structures in our data set (a) Different horizontal period or they have the same horizontal period but are not aligned, (b) Same vertical period, different horizonal periods. (c) Same vertical period, different horizonal periods. The second column can belong to either structure.

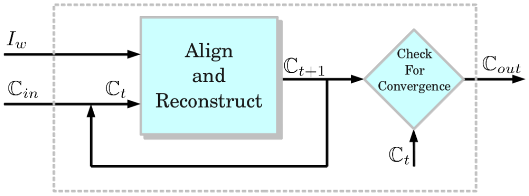


Figure 4. Overview of the Iterative Alignment and Reconstruction module (Sec. 6).

6. Iterative Alignment and Reconstruction

The next step in our pipeline is a module which, after being initialized with $\mathbb{C}_0$, it iteratively attempts to align the elements of this set and also to add new ones based on the local periodicity of the image I_w. The inputs of this module will be $\mathbb{C}_0$ and I_w and the output will be the aligned and reconstructed centers $\mathbb{C}_T$, where $\mathbb{C}_t$ is the result derived from iteration $t = 0, 1, ..., T$. In order to understand the notion of local window periodicity in a dataset of building facades, in Fig.3 we present some synthetic facades demonstrating three cases that occur frequently and consist of multiple periodic regions.

The iterative procedure is visualized in Fig.4 and the details of the implementation of the single step module are presented in the next subsections and are summarized in Fig.5. When we will refer to a single iteration, for the sake of simplicity, the input set of centers will be denoted by $\mathbb{C}$ and the output by $\mathbb{C}'$. The elements of these sets will be denoted in the rest of the paper by 2D vectors $\mathbf{c}_k$, for $k = 1, ..., |\mathbb{C}|$ and $\mathbf{c}'_k$, for $k = 1, ..., |\mathbb{C}'|$.

6.1. Kernel Image Generation

Having the set $\mathbb{C}$, we would also like to have an image representation of this set such that it will have large intensities in pixels that correspond to the centers $\mathbf{c}_k \in \mathbb{C}$ and will gradually have smaller intensities in pixels which are located further from any $\mathbf{c}_k$. Hence, each pixel will be assigned a value which corresponds to the proximity of a center. The reasons for the generation of this image will become obvious in the next section, when we will try to assign lattices to each one of $\mathbf{c}_k$ as a measure of the local periodicity.

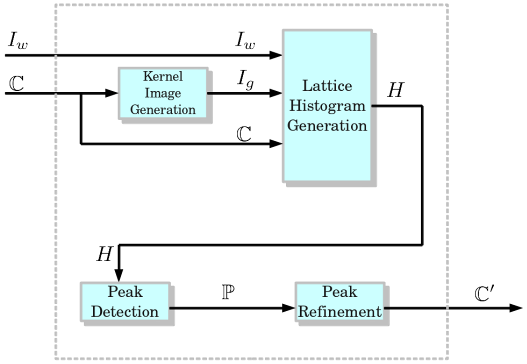


Figure 5. Diagram of a single step of Alignment and Reconstruction (Sec. 6)

.

Hence, we create a 2D image I_g of size $N \times M$, equal to the size of I_w, with 2D Gaussian kernels centered at $\mathbf{c}_k$ and intensity from 0 to 1. The σ parameter is $1.5 * w$ for the horizontal direction and $1.5 * h$ for the vertical direction, where w and h represent the width and the height of each window. For the first iteration, w and h are given from the bounding boxes of the regions of I_w (see Sec. 5), whereas in the rest of the iterations, as new centers might be created, we cannot use this information and we set $w = 2r$ and $h = 2r$. The r parameter will be used to tune other modules as well and is a heuristically chosen typical value ($r = 100cm$) for window width. It also represents the minimum distance between two window centers.

6.2. Lattice Histogram Generation - Voting Scheme

This section will analyze the generation of a *lattice histogram* (H) through a procedure which we call *lattice voting*. The reader can refer to Algorithm 1 and Fig.6 for a summary.

Before we proceed to the rest of the analysis of our algorithm, it is essential to define the *rectangular lattice*, which is a structure that is widely used in this work. A rectangular lattice $\mathbf{L} = \mathbf{L}(\mathbf{g}, Q, R, \mathbf{l}_{11})$ with a *generator* $\mathbf{g} = (p_x, p_y)$, $p_x, p_y \geq 0$ and dimensions $Q \times R$ is defined as follows:

$$\begin{aligned} \mathbf{L} = \Big\{ \mathbf{l}_{ij} &= \mathbf{l}_{11} + \Big((i-1)p_x, (j-1)p_y \Big), \\ i &= 1, 2, ..., Q, \\ j &= 1, 2, ..., R \Big\} \end{aligned} \tag{1}$$

In other words, it is a set of $Q \times R$ 2D points arranged in a regular matrix, where columns are spaced by p_x, rows by p_y and the element $\mathbf{l}_{11}$ is the upper left element. Each of the $\mathbf{l}_{ij} \in \mathbf{L}$ can be considered as a *seed* or *node* of the lattice.

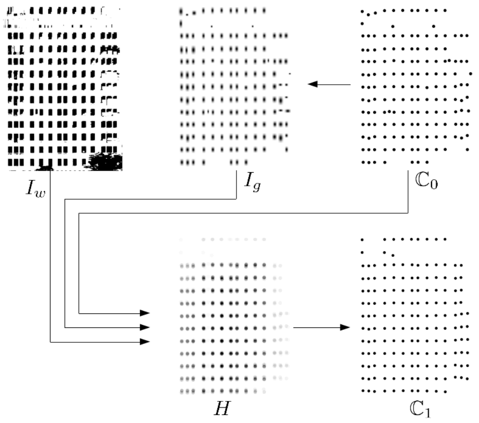


Figure 6. The variables of the Lattice Histogram Generation and their relations for the first iteration of Alignment and Reconstruction. Darker pixels show higher value. See also Fig.5.

In the cases where $p_x = 0$ or $p_y = 0$, the lattice becomes one-dimensional.

The basic idea behind lattice voting is that if we concentrate on each $\mathbf{c}_k$ and manage to generate local lattices that (a) contain this point and (b) are a good fit for the data, then we have a representation of the local periodicity of the region in which $\mathbf{c}_k$ belongs to. The aggregated information of all the lattices generated from all $\mathbf{c}_k$ will help us estimate updated locations $\mathbf{c}'_k$. We proceed by setting up a *Hough-like lattice voting scheme*, where, given I_w, I_g and $\mathbb{C}$, each window center $\mathbf{c}_k$ will vote for new centers, that correspond to the nodes of lattices that satisfy (a) and (b). The resulting histogram (H), denoted as the *Lattice Voting Space*, will have peaks in locations that will be the elements of the new $\mathbb{C}'$. Eventually, $\mathbb{C}'$ will contain points that demonstrate a better alignment than the ones in $\mathbb{C}$ and will probably include new points if there are peaks in locations where no element of $\mathbb{C}$ was close. The latter points are the *reconstructed* centers. If the assumption of underlying multiple regular structures holds, occluded or yet undetected windows will be revealed in the next iterations. The details of this procedure are organized in 4 steps which are also summarized in Algorithm 1.

Step 1 A first step towards fitting lattices in our data is to choose specific generators that agree with the local periodicity of I_w and I_g. Given a center $\mathbf{c}_k$, we find its K_T nearest centers and we create a histogram of the ${K_T}^2$ possible pairwise translations (also see [18], [9]). Each translation will vote for absolute values of x and y. The resulting histogram (generator space) is denoted by $\mathbf{Th}_k$.

We are interested in finding the minimum non-zero generator (period) of the neighborhood. So, we create projection profiles along the x and y axes of $\mathbf{Th}_k$ (adapted from [5]) and we constraint the generator space to include only the generators which have $x < 1.5x_p$ and $y < 1.5y_p$, where $x_p, y_p > 0$ are the first peaks of the corresponding profiles.

Consequently, by locating the peaks in the constrained generator space, we will end up with the candidate generators denoted by $\mathbf{g}_{kn}$, where $n = 1, 2, ..., G_k$ and G_k is the number of the peaks.

Step 2 Now, need to select each candidate generator and associate it with a lattice which will be large enough but at the same time it will fit well both the window image I_w and the kernel image I_g. This way, we are looking for lattices that are in accordance not only with the original image I_w but also with the kernel image, which might contain centers that do not exist in I_w and which also integrates the information of the proximity to a window center. We can say that I_w is a coarse representation of our structure and I_g is a more refined one. We would also like the lattice to be minimal, in the sense that if we overlay it over I_w there should be much more 1s around the nodes $\mathbf{l}_{ij}$ than between them. Such a lattice can be said that it does not "skip" windows (see $sk(\mathbf{L})$ below). This, was partially ensured by choosing minimum generators in the previous step but since the generator distribution can be arbitrary, we need to perform a second check.

In order to find the best lattice for each $\mathbf{g}_{kn}$, we exhaustively consider any possible lattice $\mathbf{L}$ that is small enough to fit inside I_w, contains $\mathbf{c}_k$ and has $\mathbf{g}_{kn}$ as its generator. All these lattices are assigned a score by the following scoring function f_1 and the one with the larger score is kept and denoted by $\mathbf{L}_{kn}$:

$$f_1(\mathbf{L}) = \alpha \times \left(\frac{iFit(\mathbf{L}) + cFit(\mathbf{L})}{2} - sk(\mathbf{L}) \right) + \beta \times sz(\mathbf{L}) \tag{2}$$

where $iFit(\mathbf{L}) = \frac{\sum_{ij} I_w(\mathbf{l}_{ij})}{QR}$ is the fraction of lattice points on window regions of I_w, $cFit(\mathbf{L}) = \frac{\sum_{ij} I_g(\mathbf{l}_{ij})}{QR}$ is the normalized sum of the intensities of I_g at the lattice points,

$$\begin{aligned} sk(\mathbf{L}) \quad = \quad & \frac{1}{2QR - Q - R} \left(\sum_{i=1}^{Q-1} \sum_{j=1}^{R} I_w \left(\frac{\mathbf{l}_{ij} + \mathbf{l}_{(i+1)j}}{2} \right) \right. \\ + \quad & \left. \sum_{i=1}^{Q} \sum_{j=1}^{R-1} I_w \left(\frac{\mathbf{l}_{ij} + \mathbf{l}_{i(j+1)}}{2} \right) \right) \end{aligned} \tag{3}$$

measures the fraction of "1" pixels of I_w in the middle of the horizontal and vertical edges of the lattice,

$$sz(\mathbf{L}) = \frac{1}{2} \left(\frac{p_y(Q-1)}{N} + \frac{p_x(R-1)}{M} \right) \tag{4}$$

is a normalized measure of the size of the lattice and $\alpha > \beta > 0$.

$iFit$ rewards lattices that represent the original image I_w and $cFit$ penalizes lattices that do not pass through the centers. Both quantities are necessary to be calculated as $iFit$ will encourage the expansion of the lattice to windows not yet represented in $\mathbb{C}$ and $cFit$ will validate reconstructed windows originally hidden in I_w. We also need to reward lattices for their size because, otherwise, minimal lattices (i.e. $Q = 0, R = 2$) will be selected. Finally, the sk quantity penalizes lattices that "skip" window regions. We chose $\alpha = 0.65$ and $\beta = 0.35$ to focus more to the fit of the lattice than to its size. However, our system is not sensitive to small variations of these values.

Step 3 At this point, each $\mathbf{c}_k$ has an associated set of lattices $\mathbb{L}_k = \{\mathbf{L}_{kn}\}$ whose elements correspond to generators $\mathbf{g}_{kn}$. For $n = 1, 2, ..., |\mathbb{L}_k|$, we calculate a second score for the lattices using the scoring function f_2:

$$f_2(\mathbf{L}) = \alpha\times\left(\frac{iFit(\mathbf{L}) + cFit(\mathbf{L})}{2}\right)+\beta\times sz(\mathbf{L})-dFM(\mathbf{L}) \tag{5}$$

where dFM (*distance from mean*) is the normalized distance between $\mathbf{g}_{kn}$ and $\frac{\sum_{n=1}^{G_k}\mathbf{g}_{kn}}{G_k}$. This scoring function instead of considering the sk value, it penalizes lattices that have generators which are outliers. Finally, we sort $\mathbb{L}_k$ in descending order based on the f_2 score.

Step 4 We choose the top K_L lattices to participate in our voting scheme. In our experiments, we chose $K_L = 2$ to reduce running time and since this choice seemed to yield satisfactory results.

At this point, *lattice voting* takes place as follows: each top lattice $\mathbf{L}$ will vote in the image space by casting $Q \times R$ votes, one for each of its nodes $\mathbf{l}_{ij}$. However, each vote does not correspond to a single bin, but to a disc of bins centered at $\mathbf{l}_{ij}$ and with a diameter equal to r (see Fig.7). This is done in order to compensate for the variation between the elements of $\mathbb{C}$ and a perfectly aligned set.

It was observed that among the K_L lattices there can be lattices that significantly deviate from the ideal local periodicity and introduce variance in the peaks of H. In order to suppress this effect, lattices with $cFit < t_{vote}$ are not allowed to vote. t_{vote} can be from 0 to 1; high values reduce the number of inaccurate lattices but also reduce the number of *reconstruction lattices* i.e. the lattices that extend to locations of no initial center attempting to create new correct centers. In our experiments, we set $t_{vote} = 0.5$.

Algorithm 1 Voting Scheme

```
for all centers $\mathbf{c}_k$, where $k = 1, ..., K_C$ do
    Find candidate generators $\mathbf{g}_{kn}$ (step 1).
    for all $\mathbf{g}_{kn}$, where $n = 1, ..., G_k$ do
        Find best lattice $\mathbf{L}_{kn}$ based on the $f_1$ score (step 2).
    end for
    Sort $\mathbf{L}_{kn}$ based on the $f_2$ score (step 3).
    Use the $K_L$ top lattices to vote in $H$ (step 4).
end for
```

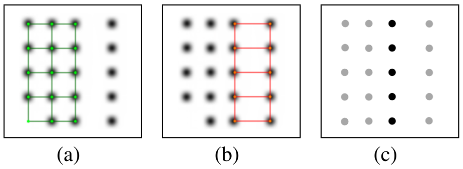


Figure 7. Illustration of *lattice voting*. (a) and (b) are different extracted lattices overlaid on I_g and (c) is the resulting histogram. The darker discs accumulate votes from both lattices. Notice the reconstruction of the lower left center.

6.3. Peak Location and Refinement

Having H, we need to locate its peaks. The nature of H allows us to achieve this by simply blurring the image and looking for pixels with an intensity higher than the ones around them. Because of the variance of the estimation (caused by voting on discs and Gaussian kernels in I_w), it is possible that the peaks belonging to one structure might have a common translational shift. In order to reduce this effect, we should find the different structures and try to align them with the centers of the previous iteration in a rigid ICP-like manner. In practice, it is sufficient to sort the points based on their x_k and split this set into K equally sized sets $\mathbb{C}'_q$. In our experiments, we chose $K = 5$. We translate each $c'_{qi} \in \mathbb{C}'_q$ by $-dT$, where:

$$dT(\mathbb{C}'_q, \mathbb{C}) = \frac{\sum_{i=1}^{|\mathbb{C}'_q|}\delta_r(\mathbf{c}'_{qi}, N_{\mathbb{C}}(\mathbf{c}'_{qi})) \parallel \mathbf{c}'_{qi} - N_{\mathbb{C}}(\mathbf{c}'_{qi}) \parallel}{\sum_{i=1}^{|\mathbb{C}|}\delta_r(\mathbf{c}'_{qi}, N_{\mathbb{C}}(\mathbf{c}'_{qi}))} \tag{6}$$

where $N_{\mathbb{S}}(\mathbf{x})$ is the nearest neighbor of $\mathbf{x}$ among the elements of set $\mathbb{S}$, $|\mathbb{S}|$ is the cardinality of $\mathbb{S}$ and

$$\delta_r(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} 1 \text{ if } \parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel < r \\ 0 \text{ otherwise} \end{cases}.$$

Eq.(6) measures the average distance (only if it is smaller than r) between each point in $\mathbb{C}'_q$ and its closest neighbor in $\mathbb{C}$.

6.4. Convergence

We check the output of the iterative procedure for convergence in order to stop and generate the output, as seen in Fig.4. For a each iteration, the following quantities are calculated:

	$\hat{dT}$	pr	re	re_v	re_p	re_o
$\mathbb{C}_0$	18 ± 1	0.910	0.738	0.912	0.392	0.009
$\mathbb{C}_T$	21 ± 1	0.894	0.810	0.919	0.633	0.237

Table 1. Evaluation of the results for centers in $\mathbb{C}_0$ (no iteration) and $\mathbb{C}_T$ (last iteration). The precision is denoted by pr. The recall can be analyzed into: overall (re), visible windows (re_v), partially occluded (re_p) and fully occluded (re_o). $\hat{dT}$ is the mean dT and is measured in cm.

$$d\mathbb{C} = \frac{\sum_{n=1}^{|\mathbb{C}'|} \| \mathbf{c}'_i - N_{\mathbb{C}}(\mathbf{c}'_i) \|}{|\mathbb{C}'|} \quad (7)$$

$$d|\mathbb{C}| = |\mathbb{C}'| - |\mathbb{C}| \quad (8)$$

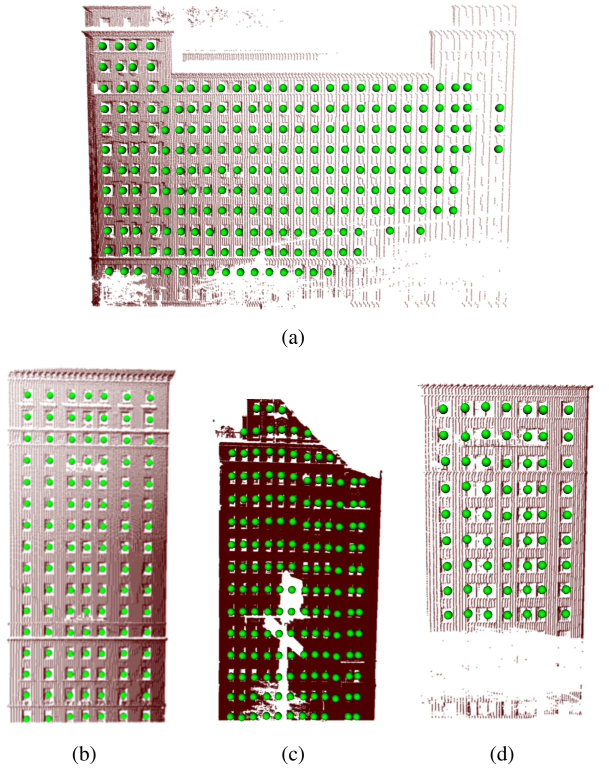
where $\| \mathbf{x} \|$ is the L_2 norm of $\mathbf{x}$. After experimentation, we tuned our process to terminate when $d\mathbb{C} < 4$ and $d|\mathbb{C}| = 0$. After the algorithm terminates, we project the 2D windows back to 3D.

7. Experiments and Performance evaluation

For our experiments, we automatically segmented 96 urban facades of various resolutions and sizes. We have acquired 3D range scans of urban scenes using a Leica ScanStation2 scanner. We compared the results of our algorithm to manually annotated window centers done through a special GUI. We denote all the annotated centers by $\mathbb{C}_A$ and the estimated centers by $\mathbb{C}_E$. The annotated centers are classified as *visible*, *partially occluded*, *fully occluded* based on the window visibility. The sets of these centers are denoted by $\mathbb{C}_{Av}$, $\mathbb{C}_{Ap}$, $\mathbb{C}_{Ao}$. We find $N_{\mathbb{C}_A}(\mathbf{c})$ (with a distance less than r) for each $\mathbf{c} \in \mathbb{C}_E$ and we define the sets $\mathbb{C}_{Ev}$, $\mathbb{C}_{Ep}$, $\mathbb{C}_{Eo}$ based on the label of $N_{\mathbb{C}_A}(\mathbf{c})$. We also define the quantity: $P_{A/E} = \sum_{i=1}^{|\mathbb{C}_A|} \delta_r(\mathbf{c}_i^A, N_{\mathbb{C}_E}(\mathbf{c}_i^A))$ as the number of detected windows. Correspondingly we can define $P_{Av/Ev}$, $P_{Ap/Ep}$, $P_{Ao/Eo}$. Thus, the precision is $pr = P_{A/E}/|\mathbb{C}_E|$ and the recall is $re = P_{A/E}/|\mathbb{C}_A|$ with an uncertainty of dT. The recalls for the window labels are: $re_i = P_{Ai/Ei}/|\mathbb{C}_{Ai}|$, $i = v, p, o$. Since the algorithm does not have knowledge of the labels of the detected windows, pr_i is not defined. The results are analyzed in Table 1.

The size of the 2D maps generated by our data is of the order of $10^5 - 10^7$ pixels. The total number of windows in our dataset is 6614 of which 4744 are labeled as visible, 1405 as partially occluded and 465 as fully occluded. The minimum number of windows per facade is 2 and the maximum 349. The running time per facade in a 3GHz pentium with 8GB RAM is 16 ± 7 mins in a C++ implementation.

The detection of windows based on region growing in I_w is considered to be the *naïve approach* (first row of Table 1). We can see that our method, although it demonstrates a similar precision than the naïve approach, it can consequently reveal more partially and fully occluded windows keeping the same recall for visible windows. These instances are hard to be detected just by region growing because in most cases no contiguous regions are formed. The difference in

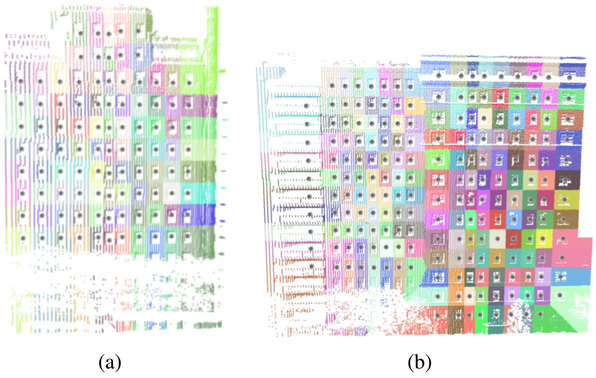

(a)

(b)

(c)

(d)

Figure 8. Results of our algorithm. Notice the reconstruction of fully occluded windows and the satisfactory performance in very low resolution regions. The wrong detections of the 5th column of (a) and the 10th one of (c) can be eliminated in a post-processing step.


(a)

(b)

Figure 9. Results of our algorithm as Voronoi regions constructed from the detected window centers (see pdf for color). Notice the reconstruction of occluded windows (lower right region of (b)) and the satisfactory performance in very low resolution regions.

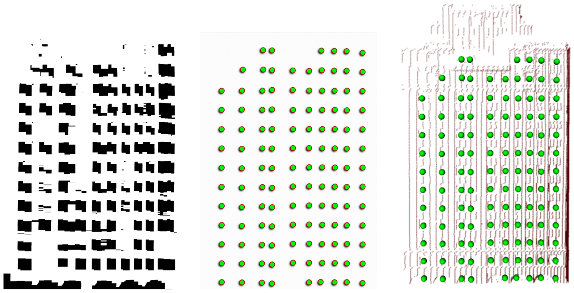


Figure 10. Low resolution example. (a) I_w, (b) final $\mathbb{C}_T$, (c) $\mathbb{C}_{3D}$ on 3D facade. Note that this is a very hard input example and we are able to detect the majority of windows.

$\hat{dT}$ is negligible, given that the GUI labeling was done with a $6cm$ accuracy. Some visual results of our experiments can be seen in Figs.8, 9 and 10. As we did not assume a single rectilinear structure, we could detect windows in examples like the one in Fig.9(b), where two facades were segmented together since they have the same major plane and zero gap between them. Also, a few problematic cases can exist, such as the one in Fig.8(a). In this case, due to big size of the structure, some lattices with relatively small $iFit$ were not adequately penalized, thus yielding a non-existent windows column. A post-processing check can eliminate such cases.

8. Conclusion

We presented an approach with strong results that detects the centers of windows in planar 3D facades of various acquisition resolutions under the assumption of the existence of regular window structures. This approach is a step towards the creation of facade descriptors which can be used in 3D registration, reconstruction, compression and scene understanding.

Acknowledgements

This work has been supported in part by the following NSF grants: IIS-0915971, CCF-0916452 and MRI CNS-0821384. We would like to thank Emmanouil Z. Psarakis for reviewing and proofreading this paper and for the very useful suggestions and discussions.

References

[1] D. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, May 2002.

[3] S. Friedman and I. Stamos. Real time detection of repeated structures in point clouds of urban scenes. *3DIMPVT*, 0:220–227, 2011.

[4] M. Jahangiri and M. Petrou. An attention model for extracting components that merit identification. ICIP'09, pages 961–964, Piscataway, NJ, USA, 2009. IEEE Press.

[5] S. C. Lee and R. Nevatia. Extraction and integration of window in a 3D building model from ground view images. In *CVPR*, volume 2, pages 113–120, June 2004.

[6] P. Muller, , G. Zeng, P. Wonka, and L. V. Gool. Image-based procedural modeling of facades. In *SIGGRAPH*, 2007.

[7] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. van Gool, and W. Purgathofer. A survey of urban reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, 2012.

[8] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Translation-symmetry-based perceptual grouping with applications to urban scenes. In *Proceedings of the 10th Asian conference on Computer vision - Volume Part III*, ACCV'10, pages 329–342, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *SIGGRAPH*, 27(3):#43, 1–11, 2008.

[10] S. Pu and G. Vosselman. Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):575–584, 2009.

[11] N. Ripperda and C. Brenner. Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. In *12th AGILE International Conference on Geographic Information Science*, 2009.

[12] C.-H. Shen, S.-S. Huang, H. Fu, and S.-M. Hu. Adaptive partitioning of urban facades. *SIGGRAPH ASIA*, 30(6):184:1–184:9, 2011.

[13] I. Stamos and P. K. Allen. Geometry and texture recovery of scenes of large scale. *Journal of Computer Vision and Image Understanding*, 88(2):94–118, 2002.

[14] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. *CVPR*, 0:2273–2280, 2011.

[15] R. Triebel, K. Kersting, and W. Burgard. Robust 3D scan point classification using associative markov networks. In *IEEE ICRA*, pages 2603 –2608, May 2006.

[16] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *ECCV*, pages 145–155, 2010.

[17] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Trans. Graph.*, 28:114:1–114:12, December 2009.

[18] P. Zhao and L. Quan. Translation symmetry detection in a fronto-parallel view. In *CVPR*, pages 1009–1016. IEEE, 2011.

[19] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen. Non-local scan consolidation for 3D urban scenes. *ACM Transactions on Graphics*, 29(4):94, 2010.