HouseCraft: Building Houses from Rental Ads and Street Views

Hang Chu Shenlong Wang Raquel Urtasun Sanja Fidler
{chuhang1122, slwang, urtasun, fidler}@cs.toronto.edu

University of Toronto

Abstract. In this paper, we utilize rental ads to create realistic textured 3D models of building exteriors. In particular, we exploit the address of the property and its floorplan, which are typically available in the ad. The address allows us to extract Google StreetView images around the building, while the building's floorplan allows for an efficient parametrization of the building in 3D via a small set of random variables. We propose an energy minimization framework which jointly reasons about the height of each floor, the vertical positions of windows and doors, as well as the precise location of the building in the world's map, by exploiting several geometric and semantic cues from the StreetView imagery. To demonstrate the effectiveness of our approach, we collected a new dataset with 174 houses by crawling a popular rental website. Our experiments show that our approach is able to precisely estimate the geometry and location of the property, and can create realistic 3D building models.

Keywords: 3D reconstruction, 3D scene understanding, localization

Fig. 1: We exploit rental ads and a small set of (wide-baseline) Google's StreetView images to build realistic textured 3D models of the buildings' exterior. In particular, our method creates models by using only the (approximate) address and a floorplan extracted from the rental ad, in addition to StreetView.

1 Introduction

Significant effort is being invested into creating accurate 3D models of cities. For example, Google and other map providers such as OpenStreetMap are augmenting their maps with 3D buildings. Architects craft such models for urban/property planning, and visualization for their clients. This process typically

involves expensive 3D sensors and/or humans in the loop. Automatically creating accurate 3D models of building exteriors has thus become an important area of research with applications in 3D city modeling, virtual tours of cities and urban planning [1–3]. The problem entails estimating detailed 3D geometry of the building, parsing semantically its important facade elements such as windows and doors, and precisely registering the building with the world's map.

Most existing approaches to 3D building estimation typically require LIDAR scans from either aerial [4, 5] or ground-level views [2], or video scans [1]. While these approaches have shown impressive results [1, 2, 6], their use is inherently limited to the availability of such sensors. In this paper, our goal is to enable a wider use, where the user can easily obtain a realistic model of her/his house by providing only an approximate address and a floorplan of the building.

Towards this goal, we exploit rental ads which contain both the property's address as well as a floor plan. We convert the address to a rough geo-location, and exploit Google's Geo-Reference API to obtain a set of StreetView images where the property's exterior is visible. A floorplan provides us with an accurate and metric outline of the building's exterior along with information about the position of windows and doors. This information is given in the footprint of the building, and typically the vertical positions (along the height) are not known.

Our approach then reasons jointly about the 3D geometry of the building and its registration with the Google's StreetView imagery. In particular, we estimate the height of each floor, the vertical positions of windows and doors, and the accurate position of the building in the world's map. We frame the problem as inference in a Markov random field that exploits several geometric and semantic cues from the StreetView images as well as the floorplan. Note that the StreetView images have a very wide baseline, and thus relying on keypoint matching across the views would result in imprecise estimation. In our model, we exhaustively explore the solution space, by efficiently scoring our projected building model across all views. Our approach is thus also partially robust to cases where the building is occluded by vegetation.

To demonstrate the effectiveness of our approach, we collected a new dataset with 174 houses by crawling an Australian rental website. We annotated the precise pose of each house with respect to Google's StreetView images, as well as the locations of windows, doors and floors. Our experiments show that our approach is able to precisely estimate the geometry and location of the property. This enables us to reconstruct a realistic textured 3D model of the building's exterior. We refer the reader to Fig. 1 for an illustration of our approach and an example of our 3D rendering. Our dataset and source code will be made available at http://www.cs.toronto.edu/housecraft.

2 Related work

Interactive image-based modeling: Interactive modeling of buildings use one or more images as a guide to build 3D models. In the seminal work of [7], users are required to draw edges over multi-view images and the architectural

3D model is then built by a grammar of parameterized primitive polyhedral shapes. Sinha et al. [3] proposed an interactive system for generating textured 3D architectural models by sketching multiple photographs. Camera poses are recovered by structure from motion and 3D models are constrained by vanishing point detection. We refer the reader to [8] for a more systematic literature review.

Automatic 3D building modeling: Researchers also attempted to tackle the problem in a fully automatic way. Many approaches made use of LIDAR aerial imagery [4,5] for this purpose. In our review, we focus on approaches that exploit ground-level information. The general idea is to utilize prior knowledge to constrain the target 3D models, such as parallelism, orthogonality, piece-wise planar and vanishing point constraints [1, 9–12]. These methods either rely on dense point clouds reconstructed from multiview stereo [1,11] or line segment [9] features that guide the architectural reconstruction. However, line segment based methods are not robust to clutter and occlusion, while multi-view approaches rely on the input to be either a video or a set of images with relatively small motion. Unlike previous approaches, our method requires only on a very sparse set of large-baseline images (typically 3 images per house with average distance between the cameras of 16.7 meters). Furthermore, we can handle large occlusion and clutter, such as vegetation and cars in the street.

Facade parsing: Our method is also related to work on facade parsing, which aims at semantic, pixel-level image labeling of the building's facade [13–16]. Hidden structures of building facades are modeled and utilized to tackle this problem, such as repetitive windows, low-rankness and grammar constraints. Compared to these structures, our method exploits the geometry and semantic priors from the floorplan, and can thus work with a much wider range of facades.

Using floorplans for vision: Floorplans contain useful yet inexpensive geometric and semantic information. They have been exploited for 3D reconstruction [17–19] and camera localization [20–22]. However, past methods mainly utilize floorplans for indoor scenes. This is in contrast to our work which aims to exploit floorplans for outdoor image-based modeling. In [23–25], city maps with building contours are used to help localization and reconstruction. City maps differ from floorplans as they do not contain information about windows and doors. To the best of our knowledge, outdoor image-based modeling with floorplans has not been proposed in the literature before. Moreover, due to the rich information available in the floorplans, our 3D parameterized model is very compact with a relative small number of degrees of freedom.

Recently, Arth et al. [25] proposed a camera localization method using 2.5D city maps. Our work differs from theirs in two aspects. First, [25] assumes that the building height is known, e.g., from OpenStreetMaps. For residential houses considered here this information is not available. Second, it assumes that the relative camera height w.r.t. building's base is constant. In our case, reasoning about relative height is necessary due to the difference in elevation between the building and the camera.

Holistic 3D scene understanding: Holistic models reason about semantics and geometry simultaneously, resulting in performance gains. For instance, [26,

Fig. 2: The initial building's position and StreetView images. In **(a)**, blue pin shows the geo-reference result, green pins show three nearby StreetView images $S_{1,2,3}$, red lines depict the building contours parsed from the map image. **(b)**: Building's floorplan placed on the map. **(c)** shows StreetView images corresponding to $S_{1,2,3}$ in (a), respectively, overlaid with the virtual house rendered with the initial building position (from (b)) and floor, door, window heights as 1m, 2m, 3m.

.

Vertical position	base	floor	door	upper window	lower window
Mean value	-2.57m	3.33m	2.56m	0.53m	1.34m

Table 1: Average values of vertical positions computed from GT annotations.

27] perform semantic parsing and multi-view reconstruction jointly and [28–30] reason about both depth and semantic labeling from a monocular image. Our approach also reasons about semantic parsing and pose estimation jointly, but constrains the degrees of freedom via floorplan priors.

3 The SydneyHouse Dataset

We exploit rental ads to obtain our dataset. In this section we first explain data collection and analyze the statistics of our new dataset.

Data Collection: We collected our dataset by crawling an Australian real-estate website[1]. We chose this site because it contains housing information in a formatted layout. We queried the website by using the keyword "Sydney"+"House", and parsed ads for the top 1,007 search results. A rental ad is typically composed of several, mostly indoor photos, a floorplan, as well as meta data information such as the address of the property. In our work we only kept houses for which a floorplan was available (80.8%). Given the address, we obtained several outdoor street-level photos around the property via Google's

[1] `http://www.domain.com.au/`

Geo-Reference API[2]. In particular, for each rental ad we downloaded a local map around the property using Google Maps (Fig. 2(a)). We then parse the building contours from the map[3], and set the building position as the position of the closest building contour (Fig. 2(b)). This gives us a rough geo-location of the rental property. We then collected Google StreetView images around this location, where we set the camera orientation to always point towards the property of interest. Fig. 2(c) shows that although the building matches the map very well, the property does not align well with the images. Obtaining a more accurate alignment is the subject of our work. We finally discarded ads for which the houses could not be geo-referenced or were not identifiable by the annotators in StreetView imagery (65.3%). Our final *SydneyHouse* dataset contains 174 houses located in different parts of Sydney. We show examples for house selection in our dataset in the supplementary material.

Ground-truth Annotation: We annotated each house with the vertical positions (e.g., floor height) and its precise location with respect to the geo-tagged imagery. We developed a WebGL based annotation tool that constructs the 3D model of the house given a set of vertical positions. The tool visualizes the projection of the 3D model onto the corresponding StreetView images. The in-house annotators were then asked to adjust the properties until the 3D model is best aligned with the streetview imagery. For each house, we annotate the 1) vertical positions, namely the building's foundation height (w.r.t. camera), 2) each floor's height as well as 3) the door heights, and 4) windows vertical starting and ending positions. We also annotate the floorplan by specifying line segments for walls, windows, and doors, as well as building orientation and scale. On average, the total annotating time for each house was around 20 minutes.

Statistics: Our dataset consists of 174 houses. On average each house has 1.4 levels, 3.2 windows, 1.7 doors. Average values of vertical positions are in Table 1.

4 Building Houses from Rentals Ads and Street Views

In this section, we show how to create a 3D model of the building exterior given a floor plan and a set of wide-baseline StreetView images. We start by describing our parametrization of the problem in terms of the random variables denoting the building's position and vertical positions. We cast the problem as energy minimization in a Markov Random Field, where inference can be performed efficiently despite a large combinatorial solution space.

4.1 Parameterization and Energy Formulation

Given several geo-tagged StreetView images of a house, $\mathcal{I} = \{\mathcal{I}_i\}_{i=1}^{N}$, and a floorplan $\mathcal{F}$ extracted from the rental ad, our goal is to jointly estimate the 3D layout of the house as well as its accurate geo-location.

[2] `https://www.google.com/maps`

[3] In particular, we use a processing pipeline consisting of color thresholding, connected component analysis, corner detection, and sorting by geodesic distance.

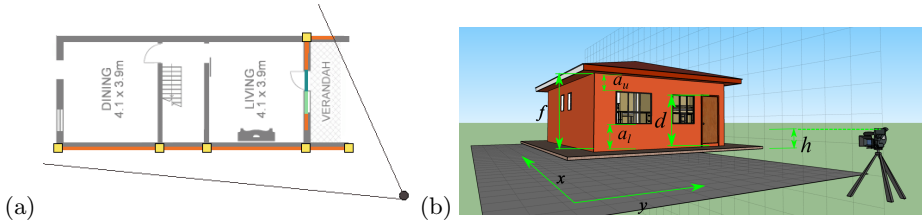

Fig. 3: Visibility and parametrization: (a) shows a floorplan and the visible parts to a given camera, (b) depicts our parametrization of the building and its pose.

A floorplan contains information about the number of floors, the dimensions of the footprint of each floor, and the (2D) location of doors and windows in the footprints. In order to lift the floorplan to 3D, we need to estimate the height of each floor, the building's foundation height as well as the vertical position of each door, window, and possibly a garage gate. Let h be the building's foundation height, where h=0 means that the building sits on a plane having the same height as the camera. Here, h thus simply encodes the vertical offset of the building's support surface from the camera. We parameterize all floors with the same height, which we denote by f. We also assume that all doors (including the garage gates) have the same height, which we denote by d. Further, let $\mathbf{a} = \{a_u, a_l\}$ be the window's vertical starting and ending position.

Our initial estimate of the house's location is not very accurate. We thus parameterize the property's true geolocation with two additional degrees of freedom (x, y), encoding the (2D) position of the house in the map. Note that this parameterization is sufficient as projection errors are mainly due to poor geolocation in Google Maps and not because of inaccurate camera estimates in StreetView. We confirmed this fact while labeling our dataset. Fig. 3 visualizes our full parametrization.

Let $\mathbf{y} = \{x, y, h, f, d, \mathbf{a}\}$ be the set of all variables we want to estimate for a house. We formulate the problem as inference in a Markov random field, which encourages the projection of the 3D model to match the image edges, semantics and location of doors and windows in all images. Furthermore, we want to encourage the building to be salient in the image, and its appearance to be different than the one of the background. Our complete energy takes the following form:

$$\begin{aligned} E(\mathbf{y}; \mathcal{I}, \mathcal{F}) = E_{\text{edge}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) + E_{\text{obj}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) + E_{\text{seg}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) \\ + E_{\text{sal}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) + E_{\text{app}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) \end{aligned} \tag{1}$$

We note that given a building hypothesis $\mathbf{y}$, our energy scores its projection in the set of StreetView images. Thus, in order to properly score a hypothesis, we need to reason about the visibility of the walls, windows, etc. We compute the visibility with a standard exact 2D visibility reasoning method [31]. Fig. 3(a) shows example of visibility reasoning. We refer the reader to suppl. material for details. In the following subsection, we describe the potentials in more detail.

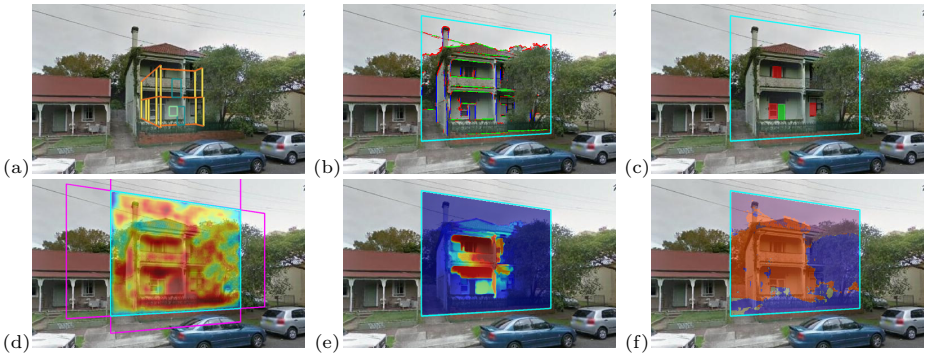

Fig. 4: Feature visualization. (a) Initial building position. All features are computed for a region around the initial building (cyan box in (b)). (b) Detected vertical (blue), horizontal (green), otherwise orientated (red) edges. (c) Detected windows (red) and doors (green). (d) Foreground color potential with foreground and background color sampled from cyan/magenta boxes. (e) Saliency-based foreground segmentation. (f) Semantic segmentation with *building* (red), *sky* (purple), *occlusion* (blue), *other* (yellow).

4.2 Potentials in our Energy

Edge Potential: This term encodes the fact that most building facade, window, door and floor boundaries correspond to image edges. We thus define

$$E_{\text{edge}}(\mathbf{y};\mathcal{I},\mathcal{F}) = \sum_{i=1}^{N} \mathbf{w}_{\text{edge}}^{T} \phi_{\text{edge}}(\mathbf{y};\mathcal{I}_i,\mathcal{F})$$

where N is the number of StreetView images, $\mathbf{w}_{\text{edge}}$ is a vector of learned weights and ϕ_{edge} is a feature vector encoding the distances between the edges of our projected hypothesis and the image edges. We compute distances for different types of building edges, i.e., *vertical*, *horizontal*, and *all*. When computing the potential, we take into account visibility and exploit distance transforms for efficiency. We use up to four different features per edge type, corresponding to different thresholded distance transforms (i.e., 0.1m, 0.2m, 0.4m). For each edge type, the potential then sums the value of the different distance transforms along the projected visible edges. In particular, we use structured edge [32] to extract the orientated edge map for each image. Fig. 4(b) shows an example.

Object Potential: To compute the object potential, we first run an object detector for three classes, i.e. *doors*, *windows*, and *garage gates*, on StreetView images. This energy term is then designed to encourage agreement between the projection of the model's visible doors, windows, and garage gates, and the detection boxes. We thus define

$$E_{\text{obj}}(\mathbf{y};\mathcal{I},\mathcal{F}) = \sum_{i=1}^{N} \mathbf{w}_{\text{obj}}^{T} \phi_{\text{obj}}(\mathbf{y};\mathcal{I}_i,\mathcal{F})$$

	Proposal		Detection			
	precision	recall	precision	recall	F1 score	overall acc.
window	10.11%	63.04%	34.65%	76.11%	47.62	84.10%
door	2.32%	32.50%	4.41%	39.29%	7.93	76.92%
garage	7.28%	49.43%	32.14%	81.82%	46.15	90.87%

Table 2: Object detection performance. Accuracy reported at each stage without taking into account previous errors (objects missed by proposals are excluded for detection).

where N is the number of images, $\mathbf{w}_{\text{obj}}$ is a vector of weights and ϕ_{obj} is a feature vector that encodes agreement for each object type. In particular, ϕ_{obj} simply counts the number of pixels for which the projected (visible) object hypothesis and the image detection box agree. We additionally use counts of pixels that are inside the projected object hypothesis, but are not in any detection box, and another count for pixels contained in the detection boxes but not contained inside the object hypothesis. Our feature vector is thus 9-dimensional (three classes and three counting features for each class). We refer the reader to Fig. 4(c) for an illustration. Note that these pixel counts can be computed efficiently with integral geometry [33], a generalization of integral images to non axis-aligned plane homographies. In our work, we rectify the image using the known homography to frontal-parallel view, so that the grid for integral geometry is axis-aligned, thus simplifying the implementation.

To detect doors, windows and garage gates in the StreetView images we use a pipeline similar to RCNN [36]. We first use edgebox [37] to generate object proposals (most objects are rectangular in the rectified view). In particular, we use only 10 object proposals since a house typically have multiple windows/doors, successfully detecting a few of them is sufficient for our task. We train a convolutional neural network on the region proposals for each class independently, by fine-tuning AlexNet [38] pre-trained on ImageNet [39], which is available in the Caffe's [40] model-zoo. Fig. 4(c) shows an example.

Segmentation Potential: This term encodes the fact that we prefer house configurations that agree with semantics extracted from StreetView images. Towards this goal, we take advantage of SegNet [41] and compute semantic segmentation for each StreetView image in terms of four classes: *sky*, *building*, *occlusion* and *other*. We define all categories that can occlude a building as *occlusion*, i.e., tree, pole, sign and vehicle. We then define our segmentation potential as

$$E_{\text{seg}}(\mathbf{y}; \mathcal{I}, \mathcal{F}) = \sum_i \mathbf{w}_{\text{seg}}^T \phi_{\text{seg}}(\mathbf{y}; \mathcal{I}_i, \mathcal{F})$$

where $\mathbf{w}_{\text{seg}}$ is a vector of weights and ϕ_{seg} is a feature vector. Each dimension of ϕ_{seg} counts the number of pixels inside the projected building region, that were labeled with one of the categories by the segmentation algorithm. We expect the learning algorithm to learn that the weight for building is positive and the weight for sky and other is negative. Similar to the object potential, this term can be

	xy/m	IOU	h/cm	f/cm	d/cm	a_l/cm	a_u/cm
random	9.07	21.04%	102.8	49.8	45.6	47.0	55.9
box-reg [34, 35]	6.68	33.31%					
google	5.01	43.46%					
ours	**2.62**	**68.29%**	**49.7**	**43.1**	**14.1**	**36.9**	**33.6**

Table 3: Mean errors of our approach and the baselines. h, f, d, a_l, and a_u denote heights for camera, floor, door, and window, respectively.

efficiently computed via 2D integral geometry in a rectified frontal-parallel view. Fig. 4(d) shows an example.

Saliency potential: This term encourages the building facade to correspond to salient objects in the scene. In particular, we use [42] to compute a per-pixel saliency score (example depicted in Fig. 4(e)). We then compute our potential by simply summing the salient scores inside the projected building's region. Note that as before we can compute this potential in constant time using integral accumulators.

Appearance potential: This term encourages the image region corresponding to the projected building's facade to have color different than the background. We sample the foreground and background color from image region around, and sufficiently far away, from the initial building position, where background normally corresponds to sky, pavement, grass, and other buildings. We compute per-pixel foreground probability with a kernel density estimator. Our potential is obtained by summing the foreground probability inside the projected facade. Fig. 4(f) shows an example.

4.3 Efficient Projection of Building Hypotheses

As shown in Fig. 3(b), given a configuration $\mathbf{y}$, we can lift the floorplan $\mathcal{F}$ to 3D, and project it onto each StreetView image $\mathcal{I}_i$ given the known camera parameters (given by Google's API). Since the camera poses are fixed (we are "moving" the building relative to the camera), we can re-write the model's translation to point $\mathbf{z}$ as: $HK[R|t]\mathbf{z} = HK[R|t](\mathbf{z}_0 + x\triangle\mathbf{z}_x + y\triangle\mathbf{z}_y)$, where H is a homography that rectifies the image to the frontal-parallel view. Here $\mathbf{z}_0$ is a chosen initial point, $(\mathbf{z}_x, \mathbf{z}_y)$ is a chosen discretization of the search space, and (x, y) represents the coordinate of our hypothesis in this space. Since $\mathbf{z}_0$ and $(\mathbf{z}_x, \mathbf{z}_y)$ are fixed for all hypotheses, we can pre-compute $HK[R|t]\mathbf{z}_0$, $HK[R|t]\mathbf{z}_x$, and $HK[R|t]\mathbf{z}_y$, allowing us to avoid matrix multiplication when projecting a new hypothesis $\mathbf{z}$.

4.4 Inference

We perform inference by minimizing the energy in Eq. (1) with respect to $\mathbf{y}$. For random variables corresponding to vertical building dimensions, we discretize our solution space by learning a set of prototypes by independently clustering

range/m	gt%	time/s	xy/m	h/cm	f/cm	a_l/cm	a_u/cm
5×5	19.0%	**6.22**	3.37	58.1	41.7	41.2	**28.4**
10×10	62.1%	8.58	2.70	**44.0**	43.3	41.5	29.6
15×15	88.5%	12.12	2.66	49.2	42.0	37.1	33.3
20×20	96.6%	19.05	**2.62**	49.7	43.1	36.9	33.6
25×25	**97.7%**	27.16	2.67	46.6	**40.4**	**35.1**	34.2

Table 4: Impact of the size of the search range on the accuracy of the method. Columns from left to right: xy search range, percentage of ground truth that is within the search range, inference time per house, average error for each variable.

quant. thres./m	time/s	xy/m	h/cm	f/cm	a_l/cm	a_u/cm
0.20	19.05	2.62	**49.7**	43.1	36.9	33.6
0.25	16.28	**2.34**	45.7	41.4	26.8	**31.3**
0.35	**12.58**	2.81	51.5	**34.0**	**26.8**	31.9

Table 5: Impact of the height quantization on inference time and accuracy.

the output space for each random variable. We refer the reader to the suppl. material for a detailed description of our discretization. It is worth noting that our energy evaluation can be done very efficiently, since all the energy potentials can be computed via integral accumulators. This allows us to perform inference via exhaustive search over the combinatorial space of all prototypes. On average we can evaluate 20.7k candidates per second on a CPU.

4.5 Learning

We learn the parameters of the model with structured-SVMs [43], using the parallel implementation of [44]. We compute the loss function as the sum of loss functions across all StreetView images. For each individual image, we compute the loss as the intersection-over-union (IOU) between the ground-truth segmentation (implied by the ground-truth layout of the building projected into the image) and the segmentation corresponding to the model's projection. We weigh each class (i.e., building, window, doors/gates) differently. We estimate these weights as well as the slack rescaling term c via cross-validation.

5 Experiments

We first provide implementation details, and then evaluate our approach on our newly collected dataset.

Implementation details. For the building's position, we use a search range of 20m×20m, discretized by 0.25m. We use k-means clustering with a cluster variance of 0.2m for our variable-wise clustering, yielding 5 prototypes for h, 3

edge	obj	other	xy/m	h/cm	f/cm	a_l/cm	a_u/cm
✓			6.42	63.4	47.1	46.7	40.3
	✓		7.24	83.3	52.5	38.0	45.4
		✓	3.50	79.5	41.2	46.7	40.3
✓	✓		5.93	59.7	44.7	37.0	40.9
✓		✓	2.84	53.9	**40.8**	46.7	40.3
	✓	✓	3.18	60.6	46.4	37.1	**33.2**
✓	✓	✓	**2.62**	**49.7**	43.1	**36.9**	33.6

Table 6: Ablation study of different types of energy terms. The *other* potential includes segmentation, saliency, and appearance. Best result is achieved with all potentials.

for f, 1 for d, 2 for a_l, and 2 for a_u. Thus in total, the number of unique solutions is $81 \times 81 \times 60$, which is roughly 0.4 million possible states.

Evaluation. We evaluate our approach on our SydneyHouse dataset. We conduct 6-fold evaluation, where for each test fold, we use 4 folds for training and 1 fold for validation to choose the hyper-parameters. We use grid search to choose the hyper-parameters over the space $c \in \{2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}\}$, and $\alpha' = \{0.5, 1, 2\}$, which is the ratio of object IOU to facade IOU in the task loss.

We compare our approach with three baselines. In the first baseline, we randomly generate the building position, with the same xy search range and discretization. We denote this baseline as *random*. In the second baseline, we feed the frontal-parallel facade image to Inception Network [34] and extract features from both global pooling and fully connected layers. We then perform box regression as in [35] to find the optimal building's bounding box, and choose the regularization parameter that yields best results. We use the new box to obtain the building position xy. This baseline is referred to as *box-reg*. In the third baseline referred to as *google*, we obtain building position xy by placing the floorplan to best overlap with the building contour on Google Maps. In all baselines, vertical positions h, f, d, a_u, and a_l are obtained by randomly selecting from the training set. Baselines are repeated 1000 times to get the average performance.

Quantitative results: As shown in Tab. 3, the box-regression baseline achieves better building position than the random baseline. However, its performance is still poor, due to limited number of training samples. Our method outperforms all baselines by a large margin. To better demonstrate our method's advantage, we also list the frontal facade IOU with the ground truth in Tab. 3. Note that our approach significantly improves the overlap with the ground truth. For the *google* baseline, we also tried setting vertical dimensions as dataset average and found our method still outperform the baseline significantly.

Object Detection: As shown in Tab. 2, object proposals detect a fraction of all objects. This is due to the fact that in our multi-view setup, only a few views face the frontal facade perpendicularly, and the rectified images are skewed and blurred in the other views. Doors are more difficult to detect than windows and garage gates as many doors are covered by the porch. Note that we report

Fig. 5: Qualitative comparison. From left to right: input StreetView image, baseline method, our approach, texture-mapped 3D house model using our result.

precision/recall of each stage without taking into account previous errors, i.e., objects that are missed by the proposal are excluded for detection.

Fig. 6: Our approach demonstrates robustness when the house is occluded. Left: original images (two for each house). Right: our results.

Impact of search space discretization: We study the impact of the size of the search range for the building's xy position in Tab. 4. There is a trade-off between efficiency and efficacy. As we increase the search range, the percentage of ground-truth samples located within the search range increases, and building position accuracy also increases. However, above a certain range the accuracy starts to drop since more ambiguous samples are included. Tab. 5 shows the impact of different quantization thresholds for the height variables. Supplementary material investigates more discretization choices.

Ablation study: We perform an ablation study of our approach in Tab. 6. Notice that incorporating more potentials increases the accuracy of our approach. Specifically, we can see that *other* potential, which includes segmentation, saliency, and appearance, helps to estimate the building position xy the most. The *edge* potentials are more useful for estimating the building foundation height h. The best result is achieved when combining all potentials.

Qualitative results: Fig. 5 shows a few qualitative results of our approach and the *google* baseline algorithm. It can be seen that the baseline cannot localize the house precisely in the image due to the mapping and geo-referencing errors. In contrast, our approach is able to provide accurate xy as well as vertical positions (floor/base heights, vertical window and door positions). Fig. 6 further shows four examples with partial occlusions caused by trees in one or two viewpoints. Despite occlusion, our approach is still able to accurately estimate the building position and vertical positions.

Fig. 7: Failure modes. Left: original images. Right: our results.

Failure modes: Fig. 7 shows a few failure modes. In property 1, our approach fails because the initial building position is too noisy. The ground truth is 16.3m from the initial position, which exceeds our 20×20 search range. Property 2 shows another difficult case, where the building is heavily occluded in the second and third view, the facade has similar color to the sky, and many non-building edges exist. In this case, our method still estimates the building xy position correctly, but fails to estimate the vertical positions.

Timing: We report the efficiency of our method. Computing detection and segmentation features is done on a GPU, while the rest of code is all executed in CPU. Our CPU implementation uses Matlab without parallelization. Training our model takes around 20 minutes each fold. In inference, our approach takes 19.05s in total per house, which includes 3.41s for computing the image features, 8.00s for rendering all configurations and 7.64s for inference. Note that in our case both rendering and inference are highly parallelizable, thus allowing for high speed-ups with a more sophisticated implementation.

6 Conclusion

In this paper, we proposed an approach which exploits rentals ads to create realistic textured 3D models of building exteriors. In particular, the property's address is employed to obtain a set of wide-baseline views of the building, while the floor plan is exploited to provide a footprint of the building's facade as well as the location on the floor of doors and windows. We formulated the problem as inference in a Markov random field that exploits several geometric and semantic cues from the StreetView images as well as the floorplan. Our experiments showed that our approach is able to precisely estimate the geometry and location of the property, and can create realistic 3D models of the building exterior.

References

1. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. In: ACM Transactions on Graphics (TOG). (2009)

2. Pylvanainen, T., Berclaz, J., Korah, T., Hedau, V., Aanjaneya, M., Grzeszczuk, R.: 3d city modeling from street-level data for augmented reality applications. In: 3DIM/3DPVT. (2012)
3. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3d architectural modeling from unordered photo collections. In: ACM Transactions on Graphics (TOG). (2008)
4. Verma, V., Kumar, R., Hsu, S.: 3d building detection and modeling from aerial lidar data. In: CVPR. (2006)
5. Zebedin, L., Bauer, J., Karner, K.F., Bischof, H.: Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In: ECCV. (2008)
6. Wang, L., Neumann, U.: A robust approach for automatic registration of aerial images with untextured aerial lidar data. In: CVPR. (2009)
7. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: SIGGRAPH. (1996)
8. Musialski, P., Wonka, P., Aliaga, D.G., Wimmer, M., Gool, L., Purgathofer, W.: A survey of urban reconstruction. In: Computer graphics forum. (2013)
9. Werner, T., Zisserman, A.: New techniques for automated architectural reconstruction from photographs. In: ECCV. (2002)
10. Dick, A.R., Torr, P.H., Cipolla, R.: Modelling and interpretation of architecture from several images. IJCV (2004)
11. Sinha, S.N., Steedly, D., Szeliski, R.: Piecewise planar stereo for image-based rendering. In: ICCV. (2009)
12. Mičušík, B., Košecká, J.: Multi-view superpixel stereo in urban environments. IJCV (2010)
13. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. In: ACM Transactions on Graphics (TOG). (2007)
14. Martinović, A., Mathias, M., Weissenberg, J., Van Gool, L.: A three-layered approach to facade parsing. In: ECCV. (2012)
15. Teboul, O., Simon, L., Koutsourakis, P., Paragios, N.: Segmentation of building facades using procedural shape priors. In: CVPR. (2010)
16. Cohen, A., Schwing, A.G., Pollefeys, M.: Efficient structured parsing of facades using dynamic programming. In: CVPR. (2014)
17. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: ICCV. (2009)
18. Cabral, R., Furukawa, Y.: Piecewise planar and compact floorplan reconstruction from images. In: CVPR. (2014)
19. Ikehata, S., Yang, H., Furukawa, Y.: Structured indoor modeling. In: ICCV. (2015)
20. Liu, C., Schwing, A.G., Kundu, K., Urtasun, R., Fidler, S.: Rent3d: Floor-plan priors for monocular layout estimation. In: CVPR. (2015)
21. Wang, S., Fidler, S., Urtasun, R.: Lost shopping! monocular localization in large indoor spaces. In: ICCV. (2015)
22. Chu, H., Ki Kim, D., Chen, T.: You are here: Mimicking the human thinking process in reading floor-plans. In: ICCV. (2015)
23. Untzelmann, O., Sattler, T., Middelberg, S., Kobbelt, L.: A scalable collaborative online system for city reconstruction. In: ICCV Workshops. (2013)
24. Strecha, C., Pylvänäinen, T., Fua, P.: Dynamic and scalable large scale image reconstruction. In: CVPR. (2010)
25. Arth, C., Pirchheim, C., Ventura, J., Schmalstieg, D., Lepetit, V.: Instant outdoor localization and slam initialization from 2.5d maps. TVCG (2015)

26. Savinov, N., Ladicky, L., Hane, C., Pollefeys, M.: Discrete optimization of ray potentials for semantic 3d reconstruction. In: CVPR. (2015)
27. Kundu, A., Li, Y., Dellaert, F., Li, F., Rehg, J.M.: Joint semantic segmentation and 3d reconstruction from monocular video. In: ECCV. (2014)
28. Wang, S., Fidler, S., Urtasun, R.: Holistic 3d scene understanding from a single geo-tagged image. In: CVPR. (2015)
29. Liu, B., Gould, S., Koller, D.: Single image depth estimation from predicted semantic labels. In: CVPR. (2010)
30. Ladickỳ, L., Shi, J., Pollefeys, M.: Pulling things out of perspective. In: CVPR. (2014)
31. Teller, S.J., Séquin, C.H.: Visibility preprocessing for interactive walkthroughs. In: SIGGRAPH. (1991)
32. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)
33. Schwing, A.G., Hazan, T., Pollefeys, M., Urtasun, R.: Efficient Structured Prediction for 3D Indoor Scene Understanding. In: Proc. CVPR. (2012)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015)
35. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. (2015)
36. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR (2014)
37. Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: ECCV. (2014)
38. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
39. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3) (2015) 211–252
40. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: ACM Multimedia. (2014)
41. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
42. Zhang, J., Sclaroff, S., Lin, Z., Shen, X., Price, B., Mech, R.: Minimum barrier salient object detection at 80 fps. In: ICCV. (2015)
43. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR (2005)
44. Schwing, A.G., Fidler, S., Pollefeys, M., Urtasun, R.: Box in the box: Joint 3d layout and object reasoning from single images. In: Proc. ICCV. (2013)