A Dynamic Programming Approach to Reconstructing Building Interiors

Alex Flint, Christopher Mei, David Murray, and Ian Reid

Dept. Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
{alexf,cmei,dwm,ian}@robots.ox.ac.uk

Abstract. A number of recent papers have investigated reconstruction under Manhattan world assumption, in which surfaces in the world are assumed to be aligned with one of three dominant directions [1,2,3,4]. In this paper we present a dynamic programming solution to the reconstruction problem for "indoor" Manhattan worlds (a sub–class of Manhattan worlds). Our algorithm deterministically finds the global optimum and exhibits computational complexity linear in both model complexity and image size. This is an important improvement over previous methods that were either approximate [3] or exponential in model complexity [4]. We present results for a new dataset containing several hundred manually annotated images, which are released in conjunction with this paper.

1 Introduction

In this paper we investigate the problem of reconstructing simple geometric models from single images of indoor scenes. These scene models can be used to distinguish objects from background in recognition tasks, or provide strong global contextual cues about the observed scene (*e.g.* office spaces, bedrooms, corridors, *etc.*). Point clouds provided by structure–from–motion algorithms are often sparse and do not provide such strong indicators. Compared to a full dense reconstruction, the approach is computationally more efficient and is less sensitive to large texture–less regions typically encountered in indoor environments.

The past few years have seen considerable interest in the Manhattan world assumption [1,2,4,3,5], in which each surface is assumed to have one of three possible orientations. Making this assumption introduces regularities that can improve the quality of the final reconstruction [3]. Several papers have also investigated *indoor* Manhattan models [4,5,6] (a sub–class of Manhattan models), which consist of vertical walls extending between the floor and ceiling planes. A surprisingly broad set of interesting environments can be modelled exactly or approximately as indoor Manhattan scenes [5]. It is with this class of scenes that this paper is concerned.

The present work describes a novel and highly efficient algorithm to obtain models of indoor Manhattan scenes from single images using dynamic programming. In contrast to point cloud reconstructions, our algorithm assigns semantic

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part V, LNCS 6315, pp. 394–407, 2010.

labels such as "floor", "wall", or "ceiling". We show that our method produces superior results when compared to previous approaches. Furthermore, our algorithm exhibits running time linear in both image size and model complexity (number of corners), whereas all previous methods that we are aware of [4,5] are exponential in model complexity.

The remainder of the paper is organised as follows. Section 2 describes previous work in this area and section 3 outlines our approach. In section 4 we pose the indoor Manhattan problem formally, then in section 5 we develop the dynamic programming solution. We present experimental results in section 6, including a comparison with previous methods. Concluding remarks are given in the final section.

2 Background

Many researchers have investigated the problem of recovering polyhedral models from line drawings. Huffman [7] detected impossible objects by discriminating concave, convex, and occluding lines. Waltz [8] investigated a more general problem involving incomplete line drawings and spurious measurements. Sugihara [9] proposed an algebraic approach to interpreting line drawings, while the "origami world" of Kanade [10] utilised heuristics to reconstruct composites of shells and sheets.

Hoiem *et al.* [11] and Saxena *et al.* [12] have investigated the single image reconstruction problem from a machine learning perspective. Their approaches assign pixel–wise orientation labels using appearance characteristics of outdoor scenes. Hedau *et al.* [6] extend this to indoor scenes, though their work is limited to rectangular box environments.

The work most closely related to our own is that of Lee *et al.* [4], which showed that line segments can be combined to generate indoor Manhattan models. In place of their branch–and–bound algorithm, our system uses dynamic programming to efficiently search *all* feasible indoor Manhattan models (rather than just those generated by line segments). As a result we obtain more accurate models, can reconstruct more complex environments, and obtain computation times several orders of magnitude faster than their approach, as will be detailed in Section 6.

Furukawa *et al.* [3] used the Manhattan world assumption for stereo reconstruction. They make use of multiple calibrated views, and they search a different class of models, so their approach is not comparable to ours.

Barinova *et al.* [13] suggested modelling outdoor scenes as a series of vertical planes. Their models bear some similarity to ours but they cannot handle occluding boundaries, and their EM inference algorithm is less efficient that our dynamic programming approach.

Felzenszwalb and Veksler [14] applied dynamic programming to a class of pixel labelling problems. Because they optimise directly in terms of pixel labels their approach is unable to capture the geometric feasibility constraints that our system utilises.

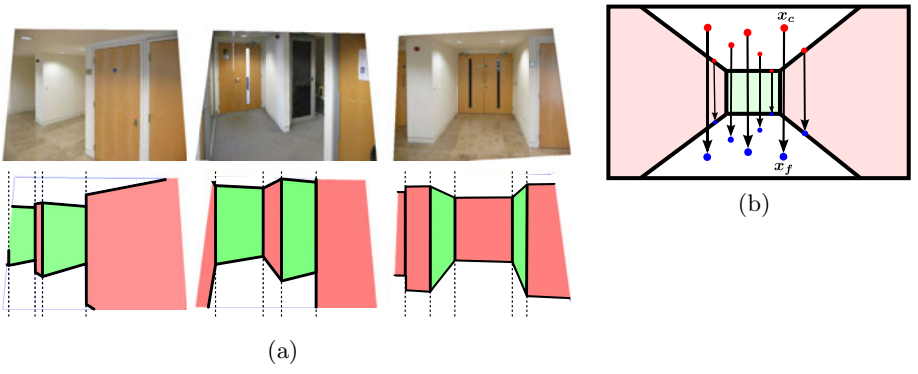


Fig. 1. (a) Three input images and the indoor Manhattan models we seek. Notice how each image column intersects exactly one wall. (b) The mapping $H_{c \to f}$ transfers points between the ceiling and floor.

3 Outline of Proposed Approach

Our goal is to reconstruct an indoor Manhattan model from a single image. Three example images and the models we seek for them are shown in Figure 1a. A perfectly uncluttered environment such as that shown in the third column of Figure 1a could be represented exactly by an indoor Manhattan model, though in general we expect to encounter clutter and our goal in such cases is to recover the boundaries of the environment in spite of this distraction. That is, we aim to completely ignore all objects within the room and reconstruct the bare room boundaries, in contrast to most previous approaches that aim to reconstruct the entire scene. This choice is due to our intention of using the models as input for further reasoning.

The Manhattan world assumption states that world surfaces are oriented in one of three mutually orthogonal directions [1]. The indoor Manhattan assumption further states that the environment consists of a floor plane, a ceiling plane, and a set of walls extending vertically between them [4]. Each wall therefore has one of two possible orientations (ignoring sign), and each corner[1] is either concave, convex, or occluding, as depicted in Figure 3. Indoor Manhattan models are interesting because they can represent many indoor environments approximately or exactly, yet they introduce regularities to the reconstruction problem that makes possible a left–to–right decomposition of the scene, on which the dynamic programming algorithm developed in this paper rests. Our approach to reconstructing indoor Manhattan environments consists of the following five steps:

1. Identify three dominant surface orientations. (Section 3.1)
2. Identify the floor and ceiling planes. (Section 3.2)

[1] We use "corner" throughout this paper to refer to the intersection of two walls, which appears as a line segment in the image.

3. Rectify vertical lines. (Section 3.3)
4. Obtain weak orientation estimates. (Section 3.4)
5. Estimate the final model. (Sections 4 and 5)

3.1 Identifying Dominant Directions

We identify three dominant directions by estimating mutually orthogonal vanishing points in the image. Our approach is similar to Kosecka̋ and Zhang [2], in which k–means clustering provides an initial estimate that is refined using EM. We assume that the vertical direction in the world corresponds to the vanishing point with largest absolute y–coordinate, which we label $\boldsymbol{v_v}$. The other two vanishing points are denoted $\boldsymbol{v_l}$ and $\boldsymbol{v_r}$.

If the camera intrinsics are unknown then we construct the camera matrix K from the detected vanishing points by assuming that the camera centre is at the image centre and choosing a focal length and aspect ratio such that the calibrated vanishing points are mutually orthogonal.

3.2 Identifying the Floor and Ceiling Planes

An indoor Manhattan scene has exactly one floor and one ceiling plane, both with normal direction $\boldsymbol{v_v}$. It will be useful in the following sections to have available the mapping $H_{c\to f}$ between the image locations of ceiling points and the image locations of the floor points that are vertically below them (see Figure 1b). $H_{c\to f}$ is a planar homology with axis $\boldsymbol{h} = \boldsymbol{v_l} \times \boldsymbol{v_r}$ and vertex $\boldsymbol{v_v}$ [15] and can be recovered given the image location of any pair of corresponding floor/ceiling points $(\boldsymbol{x_f}, \boldsymbol{x_c})$ as

$$H_{c\to f} = I + \mu \frac{\boldsymbol{v_v}\boldsymbol{h}^T}{\boldsymbol{v_v} \cdot \boldsymbol{h}} \ , \tag{1}$$

where $\mu =< \boldsymbol{v_v}, \boldsymbol{x_c}, \boldsymbol{x_f}, \boldsymbol{x_c} \times \boldsymbol{x_f} \times \boldsymbol{h} >$ is the characteristic cross ratio of $H_{c\to f}$.

Although we do not have *a priori* any such pair $(\boldsymbol{x_f}, \boldsymbol{x_c})$, we can recover $H_{c\to f}$ using the following RANSAC algorithm. First, we sample one point $\hat{\boldsymbol{x}}_{\boldsymbol{c}}$ from the region above the horizon in the Canny edge map, then we sample a second point $\hat{\boldsymbol{x}}_{\boldsymbol{f}}$ collinear with the first and $\boldsymbol{v_v}$ from the region below the horizon. We compute the hypothesis map $\hat{H}_{c\to f}$ as described above, which we then score by the number of edge pixels that $\hat{H}_{c\to f}$ maps onto other edge pixels (according to the Canny edge map). After repeating this for a fixed number of iterations we return the hypothesis with greatest score.

Many images contain either no view of the floor or no view of the ceiling. In such cases $H_{c\to f}$ is unimportant since there are no corresponding points in the image. If the best $H_{c\to f}$ output from the RANSAC process has a score below a threshold k_t then we set μ to a large value that will transfer all pixels outside the image bounds. $H_{c\to f}$ will then have no impact on the estimated model.

3.3 Rectifying Vertical Lines

The algorithms presented in the remainder of this paper will be simplified if vertical lines in the world appear vertical in the image. We therefore warp images according to the homography

$$H = \begin{pmatrix} \boldsymbol{v_v} \times \boldsymbol{e_3} \\ \boldsymbol{v_v} \\ \boldsymbol{v_v} \times \boldsymbol{e_3} \times \boldsymbol{v_v} \end{pmatrix} , \qquad \boldsymbol{e_3} = [0, 0, 1]^T . \tag{2}$$

3.4 Obtaining Weak Orientation Estimates

Our algorithm requires a pixel–wise surface orientation estimate to bootstrap the search. Obtaining such estimates has been explored by several authors [4,11,12]. We adopt the simple and efficient line–sweep approach of Lee *et al.* [4], which produces a partial labelling of the image in terms of the three Manhattan surface orientation labels (corresponding to the three Manhattan orientations in the image). We denote this orientation map $o : \mathbb{Z}^2 \rightarrow \{l, r, v, \emptyset\}$ where $\emptyset$ represents the case in which no label is assigned and l, r, and v correspond to the three vanishing points $\{\boldsymbol{v_l}, \boldsymbol{v_r}, \boldsymbol{v_v}\}$.

Note that our algorithm is is not dependent on the manner in which o is obtained; any method capable of estimating surface orientations from a single image, including the work of Hoiem [11] or Saxena [12], could be used instead.

We generate three binary images B_a, $a \in \{l, r, v\}$ such that $B_a(\boldsymbol{x}) = 1$ if and only if $o(\boldsymbol{x}) = a$. We then compute the integral image for each B_a, which allows us to count the number of pixels of a given orientation within any rectangular sub–image in $O(1)$ time. This representation expedites evaluation of the cost function described later.

4 Formulation of Reconstruction Problem

Consider the indoor Manhattan scenes shown in Figure 1a. Despite the complexity of the original images, the basic structure of the scene as depicted in the bottom row is simple. In each case there is exactly one wall between any two adjacent corners[1], so any vertical line intersects at most one wall. This turns out to be a general property of indoor Manhattan environments that arises because the camera must be between the floor and ceiling planes. Any indoor Manhattan scene can therefore be represented as a series of one or more wall segments in order from left to right.

Given the warp performed in the Section 3.3, corners are guaranteed to appear vertical in the image, so can be specified simply by an image column. Furthermore, given the mapping $H_{c \rightarrow f}$ from Section 3.2 the image location of either the top or bottom end–point of a corner (*i.e.* the intersection of the wall with the ceiling or floor respectively) is sufficient to specify both and thereby the line segment representing that corner. Without loss of generality we choose to represent corners by their upper end–point. A wall segment can then be specified

by its left and right corners, together with its associated vanishing point (which must be either $\boldsymbol{v_l}$ or $\boldsymbol{v_r}$), as illustrated in Figure 2a.

This leads to a simple and general parametrisation under which we represent an indoor Manhattan model $\mathcal{M}$ as an alternating sequence of corners and walls $(c_1, W_1, c_2, W_2, ..., W_{n-1}, c_n)$, $c_i < c_{i+1}$. Each corner c_i is the column index at which two walls meet, and each wall $W_i = (r_i, a_i)$ comprises an orientation $a_i \in \{l, r\}$, which determines whether its vanishing point is $\boldsymbol{v_l}$ or $\boldsymbol{v_r}$, and a row index r_i, at which its upper edge meets the corner to its left (see Figure 2b). Hence the upper–left corner of the i^{th} wall is (c_i, r_i), which, together with its vanishing point $\boldsymbol{v_{a_i}}$, fully specifies its location in the image. Clockwise from top–left the vertices of the i^{th} wall are

$$\boldsymbol{p_i} = [c_i, r_i, 1]^T,\ \boldsymbol{q_i} = \boldsymbol{p_i} \times \boldsymbol{v_{a_i}} \times [1, 0, -c_{i+1}]^T,\ \boldsymbol{r_i} = H_{c \to f} \boldsymbol{q_i},\ \boldsymbol{s_i} = H_{c \to f} \boldsymbol{p_i}\ . \quad (3)$$

A model $\mathcal{M}$ generates for each pixel $\boldsymbol{x}$ a predicted surface orientation $g_{\mathcal{M}}(\boldsymbol{x}) \in \{l, r, v\}$ corresponding to one of the three vanishing points $\{\boldsymbol{v_l}, \boldsymbol{v_r}, \boldsymbol{v_v}\}$. We compute $g_{\mathcal{M}}$ by filling quads corresponding to each wall segment, then filling the remaining area with the floor/ceiling label v.

Not all models $\mathcal{M}$ are physically realisable, but those that are not can be discarded using simple tests on the locations of walls and vanishing points as enumerated by Lee *et al.* [4]. The reader is referred to their paper for details; the key result for our purposes is that a model is feasible if all of its corners are feasible, and the feasibility of a corner is dependent only on the immediately adjoining walls.

4.1 Formalisation

We are now ready to formalise the minimisation problem. Given an input image of size $W \times H$ and an initial orientation estimate o, the pixel–wise cost $C_d(\boldsymbol{x}, a)$ measures the cost of assigning the label $a \in \{l, r, v\}$ to pixel $\boldsymbol{x}$. We adopt the simple model,

$$C_d(\boldsymbol{x}, a) = \begin{cases} 0, & \text{if } o(\boldsymbol{x}) = a \text{ or } o(\boldsymbol{x}) = \emptyset \\ 1, & \text{otherwise} \end{cases}\ . \quad (4)$$

The cost for a model $\mathcal{M}$ consisting of n corners is then the sum over pixel–wise costs,

$$C(\mathcal{M}) = n\lambda + \sum_{\boldsymbol{x} \in \mathcal{I}} C_d(\boldsymbol{x}, \mathcal{M}(\boldsymbol{x})) \quad (5)$$

where λ is a constant and $n\lambda$ is a regularisation term penalising over–complex models. We seek the model with least–cost

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmin}}\ C(\mathcal{M})\ . \quad (6)$$

where implicit in (5) is the restriction to labellings representing indoor Manhattan models, since only such labellings can be represented as models $\mathcal{M}$. Figure 1a shows optimal models $\mathcal{M}^*$ for three input images.

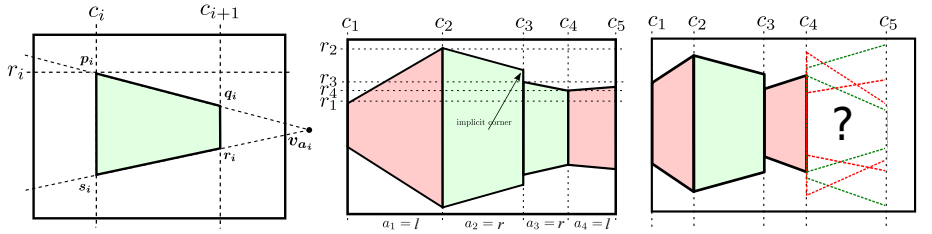


Fig. 2. (a) The row/column indices c_i, c_{i+1}, r_i, together with the vanishing point index $a_i \in \{l, r\}$ are sufficient via the homology $H_{c \to f}$ to determine the four vertices defining a wall. (b) An illustration of the model $\mathcal{M} = \{c_1, (r_1, a_1), ..., c_4, (r_4, a_4), c_5\}$. (c) A partial model covering columns to c_1 to c_4 with several feasible (green dashed) and infeasible (red dashed) wall segments.

5 Proposed Algorithm

In this section we present a dynamic programming solution to the minimisation problem posed in the previous section. We develop the algorithm conceptually first, then formalise it later.

We have already seen that every indoor Manhattan scene can be represented as a left–to–right sequence of wall segments, and every image column intersects exactly one wall segment. As a result, the placement of each wall is "conditionally independent" of the other walls given its left and right neighbours. For example, Figure 2c shows a partial model as well as several wall segments that could be appended to it. Some of the candidates are feasible (green dashes) and some are not (red dashes); however, note that once the wall segment from c_3 to c_4 is fixed, the feasibility of wall segments following c_4 is independent of choices made for wall segments prior to c_3.

This leads to a decomposition of the problem into a series of sub–problems of the form "find the minimum–cost *partial* model that terminates[2] at $\boldsymbol{x} = (c, r)$". To solve this we enumerate over all possible walls W that have top–right corner at $\boldsymbol{x}$, then for each we recursively solve the sub–problems for the partial model terminating at each $\boldsymbol{x}'$, where $\boldsymbol{x}'$ ranges along the left edge of W. This recursive process eventually reaches the left boundary of the image since $\boldsymbol{x}'$ is always strictly to the left of $\boldsymbol{x}$, at which point the recursion terminates. As is standard in dynamic programming approaches, the solution to each sub–problem is cached to avoid redundant computation. To solve the complete minimisation (6) we simply solve the sub–problems corresponding to each point on the right boundary of the image.

We now formalise the dynamic programming algorithm. Let $f_{in}(x, y, a, k)$ be the cost of a model $\mathcal{M}^+ = \{c_1, W_1, ..., W_{k-1}, c_k\}$ such that

1. $c_k = x$ (*i.e.* the model terminates at column x),
2. $W_{k-1} = (y, a)$ (*i.e.* the model terminates at row y with orientation a),

[2] A model "terminates" at the top–right corner of its right–most wall.

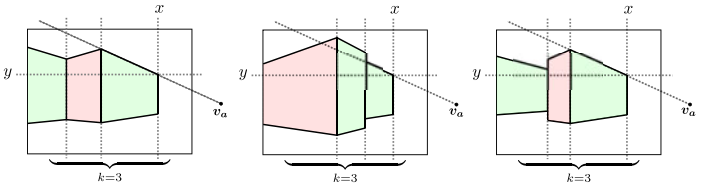


Fig. 3. Three models satisfying constraints 1–3 for the sub–problem $f_{in}(x, y, a, k)$. Only one will satisfy the least–cost constraint.

3. $\mathcal{M}^+$ is feasible, and
4. $\mathcal{M}^+$ has minimal cost among all such models.

We show in the additional material that if a model

$$\mathcal{M} = \{c_1, W_1, ..., W_{k-1}, c_k\} \tag{7}$$

is a solution to the sub–problem $f_{in}(c_k, r_k, a_k, k)$, then the the truncated model

$$\mathcal{M}' = \{c_1, W_1, ..., W_{k-2}, c_{k-1}\} \tag{8}$$

is a solution to the sub–problem $f_{in}(c_{k-1}, r_{k-1}, a_{k-1}, k-1)$. In light of this we introduce the following recurrence relation:

$$f_{in}(x, y, a, k) = \min_{x' < x,\ y', a'} \Big(f_{in}(x', y', a', k-1) + C_w \Big) , \tag{9}$$

where C_w is the cost of the wall $W = (y', a')$, computed by summing C_d over columns x' to x. The minimisation (9) is performed subject to feasibility constraints, so for each $x' < x$, only a subset of y–coordinates are considered. Since a model must have zero or more corners and a model with right–most corner at $x = 0$ does not span any part of the image, we have the boundary conditions

$$f_{in}(x, y, a, k) = \begin{cases} 0, & \text{if } x = 0 \\ \infty, & \text{if } x \neq 0 \text{ and } k < 0 \end{cases} . \tag{10}$$

Finally, the cost of the optimal model (6) is

$$C(\mathcal{M}^*) = \min_{\substack{1 \le y \le H \\ k \le K \\ a \in \{l, r\}}} \Big(f_{in}(W, y, a, k) + \lambda k \Big) . \tag{11}$$

where K is a parameter specifying the maximum model complexity and λ is the per–wall penalty.

We compute $C(\mathcal{M}^*)$ by recursively evaluating f_{in} according to (9) until we reach one of the boundary conditions (10). In line with standard dynamic programming theory we cache each evaluation to avoid redundant computation. For each cache entry we also store x', y', a' corresponding to least–cost wall identified

when evaluating (9), which allows the desired model $\mathcal{M}^*$ to be reconstructed by back–tracking once all evaluations are complete.

Complexity. Due to the caching scheme, (9) is evaluated at most once for each unique set of parameters. There are $2WHK$ possible parameters and the complexity of each evaluation is $O(W^2H)$, since the minimisation in (9) is over $O(WH)$ terms and computing each marginal cost C_w requires $O(W)$ additions[3]. The overall complexity of the basic algorithm is therefore $O(W^3H^2K) = O(L^5K)$ where $L = \max(W, H)$.

5.1 Auxiliary Sub–problems

The basic algorithm described thus far involves minimising over all pixels to the left of $\boldsymbol{x}$ for each pixel $\boldsymbol{x}$, (*i.e.* the joint minimisation over x' and y' in (9)). In the previous section we enforced feasibility by explicitly testing each (x', y') and omitting any that would lead to an infeasible model from the minimisation (9). In this section we show that by introducing auxiliary sub–problems that build feasibility into the core of the algorithm we can significantly reduce computational complexity.

We introduce three new sub–problems f_{up}, f_{down}, and f_{out}. Each is identical to f_{in} except that constraint 2 is modified as follows:

f_{out}	appending $W = (y, a)$ to M^+ would produce a feasible model.
f_{up}	$r_{k-1} \leq y$ (*i.e.* the right–most wall terminates above row y)
f_{down}	$r_{k-1} \geq y$ (*i.e.* the right–most wall terminates below row y)

Consider first the sub–problem $f_{out}(x, y, a, k)$, and suppose that the right–most wall in its solution is W'. Now W' terminates either above row y, below row y, or exactly at row y, which correspond respectively to the sub–problems f_{up}, f_{down}, and f_{in}. We also have two choices of orientation, making six possibilities in total, from which we select the one with least cost,

$$f_{out}(x, y, a, k) = \min_{a' \in \{l, r\}} \min \begin{cases} f_{up}(x, y-1, a', k) \\ f_{in}(x, y, a', k) \\ f_{down}(x, y+1, a', k) \end{cases} , \tag{12}$$

where either or both of the f_{up} and f_{down} terms are omitted if such a wall would be infeasible.

Similarly, suppose that the least–cost model that terminates at (x, y) (*i.e.* the solution to $f_{in}(x, y, a, k)$) has right–most wall W'. Now W' must have its left edge at some column $x' < x$, and the portion of the model to the left of x' must be feasible when W' is appended. Hence we have

$$f_{in}(x, y, a, k) = \min_{x' < x} \Big(f_{out}(x', y', a, k-1) + C_W \Big) , \tag{13}$$

where y' is the y–coordinate at which the line through $\boldsymbol{v_a}$ and (x, y) meets column x' and C_w is the cost of the wall $W' = (y', a')$ exactly as in (9). Note

[3] Here we use the integral images B_i.

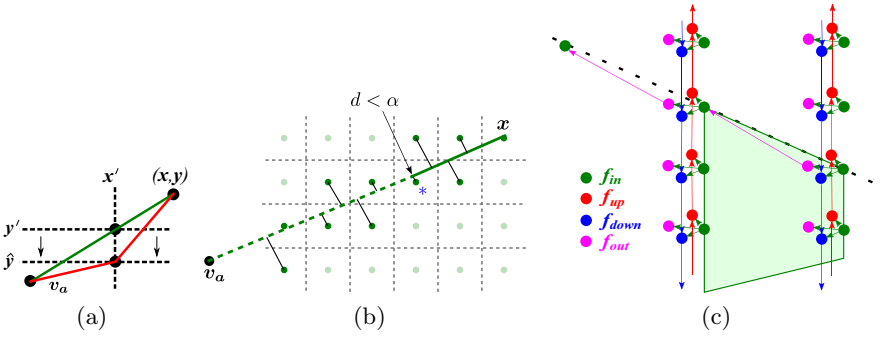


Fig. 4. (a) The bend introduced by rounding y' to $\hat{y} = \lfloor y' + 0.5 \rfloor$. (b) A line from $\boldsymbol{x}$ to $\boldsymbol{v_a}$ with the distances d to nearby pixel centres (green dots). The starred pixel is the first that satisfies $d < \epsilon$. (c) A graph in which each node represents a sub–problem and each edge is a dependence relation. Two columns are expanded; other column are omitted for brevity. The green quad is a wall corresponding to a particular pair of nodes in the graph.

that (13) consists of $O(L)$ terms whereas in the previous section (9) consisted of $O(L^2)$ terms.

Finally we may decompose the f_{up} and f_{down} sub–problems each into two cases,

$$f_{up}(x, y, a, k) = \begin{cases} \min\Big(f_{in}(\cdot), f_{up}(x, y-1, a, k)\Big), & \text{if } y \geq 1 \\ \infty, & \text{otherwise} \end{cases} \tag{14}$$

$$f_{down}(x, y, a, k) = \begin{cases} \min\Big(f_{in}(\cdot), f_{down}(x, y+1, a, k)\Big), & \text{if } y \leq H \\ \infty, & \text{otherwise} \end{cases} \tag{15}$$

The dependencies between the sub–problems are illustrated as an evaluation graph in Figure 4c.

5.2 From (L^3K) to $O(L^2K)$

Evaluating (13) remains an $O(W)$ operation due to the minimisation over x'. In this section we to reduce this to $O(1)$.

Consider the sub–problem $f_{in}(x, y, a, k)$ as formulated in the previous section. Evaluating f_{in} is like walking along each column $x-1, x-2, ..., 1$ and considering two possibilities at each step: insert a corner or continue walking. The former corresponds to evaluating $f_{out}(x', y', a, k-1)+C_w$; that is, we insert a wall between x' and x with cost C_w, then find the optimal model that occupies the remaining space to the left of x'. The latter corresponds to evaluating $f_{in}(x', y', a, k)+C_w$; that is, we find the best model that terminates at (x', y') with orientation a and extend its right–most wall to (x, y). But y' is computed by intersecting the line from $\boldsymbol{v_a}$ to (x, y) with image column x', so in general y' is not an integer. While

it is sufficient to round y' to the nearest integer $\hat{y} = \lfloor y' + 0.5 \rfloor$ when evaluating f_{out}, doing the same for f_{in} would produce a bend in the wall as shown in Figure 4a. In (13) we avoided this by evaluating f_{out} for all $x' < x$, but this is unnecessarily wasteful. We now introduce a threshold ϵ and allow $\hat{y}$ to replace y' whenever

$$|y' - \hat{y}| < \epsilon \ . \tag{16}$$

When we encounter an image column satisfying (16) we evaluate f_{in} as follows. W consider adding a corner at x' by evaluating $f_{out}(x', y', a, k-1) + C_w$ as in the previous section, then we consider the case that the wall continues past column x' by evaluating $f_{in}(x', \hat{y}, a, k) + C_w$, and we return the minimum of the two values. At this point we need not consider any further columns to the left of x' since any such consideration are already captured in the evaluation of $f_{in}(x', \hat{y}, a, k)$. Hence rather than evaluating all $x' < x$ we need only walk as far as the first x' that satisfies (16), as shown in Figure 4b. The recurrence relation for f_{in} now becomes

$$f_{in}(x, y, a, k) = \min \begin{cases} \min_{x_p \leq x' < x} \Big(f_{out}(x', y', a, k-1) + C_w \Big) \\ f_{in}(x_p, y_p), a, k-1) + C_w \end{cases} \ . \tag{17}$$

where $x_p < x$ is the closest column to x satisfying (16) and y_p is the row at that column meets the line from (x, y) to $\boldsymbol{v_a}$. Empirically we have found that even for $\epsilon = 0.01$ pixels, we always encounter some x' satisfying (16) within 20 steps from any start point.

Complexity. Evaluating each sub–problem is now an $O(1)$ operation, so the overall complexity of the algorithm is given by the total number of unique sub–problems, which is

$$O(KL^2) \ . \tag{18}$$

6 Results

We tested our system on a dataset of 634 manually annotated images of indoor scenes. To expedite annotation we collected video sequences and used structure–from–motion software to recover camera poses, allowing us to project a manually specified floor plan into each view.

In each experiment we computed the fraction of pixels for which the orientation predicted by the output model $\mathcal{M}$ agreed with the ground truth orientation. Unless otherwise specified, the parameter settings for the experiments below are $\epsilon = 0.01$, $K = 7$, $m = 4$, $\lambda = 100$. Image sizes were 640×480 pixels. We found our algorithm to be robust to all of these parameter values, as the following experiments show.

We compared our results with the branch–and–bound approach of Lee *et al.* [4]. In 138 of the images (21.7% of the dataset), their method was unable to estimate a building model as there was no appropriate pair of line segments with which to initialise their approach. In a pixel–wise evaluation their approach was

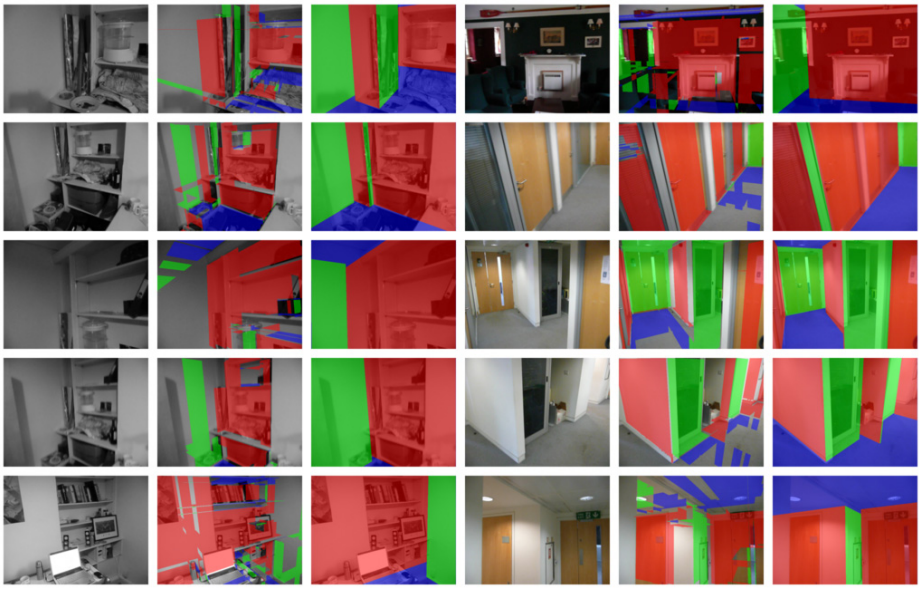


Fig. 5. Models estimated by our algorithm. Each panel contains three images: the original image, the initial orientation estimate, and the final model output by our system. Best viewed in colour.

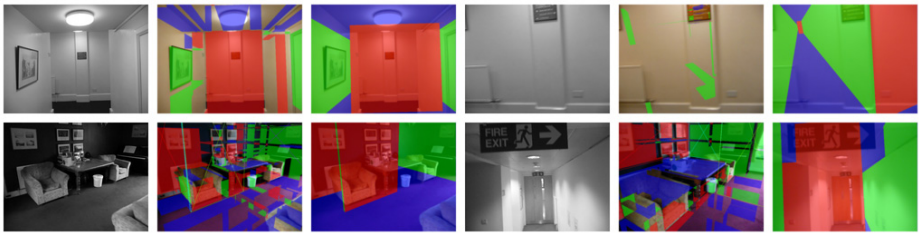
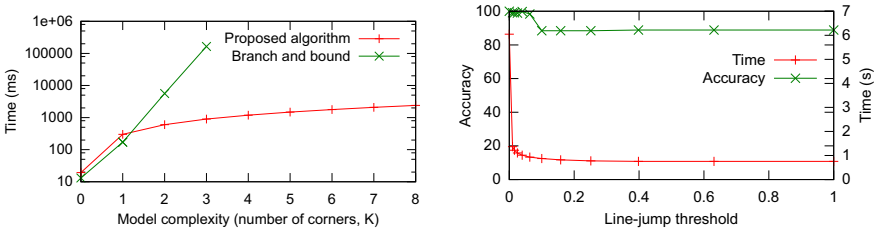


Fig. 6. Failure cases of our system. Best viewed in colour.

able to correctly label 54.3% of pixels, while our approach obtained an accuracy of 79.7%. Omitting the images for which their approach was unable to estimate a building structure, their approach obtained 68.1% accuracy. We believe that the difficulty of our dataset (many occluding objects, many images without a view of both floor and ceiling) accounts for the significantly lower performance in comparison to that quoted in [4]. Side–by–side comparisons with their approach are included in additional material.

6.1 Failure Cases

Figure 6 shows four representative failure cases of our approach. In the top–left panel the occlusion relationship between two walls is incorrectly estimated,



(a) (b)

Fig. 7. (a) Efficiency comparison with the branch–and–bound algorithm of Lee *et al.* [4] (our implementation). Their approach scales exponentially with model complexity whereas ours scales only linearly. (b) The line–jump parameter ϵ trades off accuracy (diamonds) for computation time (crosses). Accuracy is computed relative to the baseline $\epsilon = 0$. Small values of ϵ achieve significant speedup with no perceivable degradation in accuracy. Based on these results we set $\epsilon = 0.01$ in our remaining experiments.

so the more distant wall is thought to be occluding the closer wall. This is because the floor patch in the bottom centre of the image is missed in the initial orientation estimate. In the top–right panel, too few line segments are detected and the initial orientation estimate is very poor. The bottom–left panel shows an example of a chair that is wrongly identified as part of a wall. The chair is aligned with the wall behind it and this highlights the limitation of using only line segments to estimate an initial orientation estimate. The bottom–right panel shows how a deviation from the indoor Manhattan assumption causes an incorrect model to be estimated. The exit sign represents a vertical surface that does not extend from the the ceiling to the floor, which our approach is currently unable to handle.

7 Discussion

We have shown that semantically meaningful models of indoor scenes can be recovered efficiently for a range of Manhattan environments using dynamic programming. Our approach is able to model complex scenes, which would be intractable for previous methods that involved combinatorial searches in the space of models. This work represents an important increment on the state–of–the art both in terms of accuracy and efficiency.

An alternative approach might be to apply graph cuts to this problem. However, Kolmogorov and Zabih [16] showed that only regular functions (a subset of sub–modular functions) can be minimised via graph cuts, and the cost (6) is not regular because implicit in the minimisation is the hard constraint that labellings must form an indoor Manhattan model, which induces complicated dependencies between the pixels in each column. Even if an appropriate relaxation of this constraint yielded a regular cost function, applying graph cuts would entail using a technique such as α–expansion [16], which is both approximate and

non–deterministic. In contrast, our approach is exact, deterministic, and highly efficient.

Future work will investigate richer cues for obtaining the initial orientation estimates as well as a probabilistic formulation of the cost function (4).

References

1. Coughlan, J., Yuille, A.: Manhattan world: compass direction from a single image by bayesian inference. In: CVPR, vol. 2, pp. 941–947 (1999)
2. Kŏsecká, J., Zhang, W.: Video compass. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 476–490. Springer, Heidelberg (2002)
3. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Manhattan-world stereo. In: CVPR, pp. 1422–1429 (2009)
4. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: CVPR (2009)
5. Flint, A., Mei, C., Reid, I., Murray, D.: Growing semantically meaningful models for visual slam. In: CVPR (2010)
6. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: ICCV, vol. 2 (2009)
7. Huffman, D.A.: Impossible objects as nonsense sentences. Machine Intelligence 6, 295–323 (1971)
8. Waltz, D.L.: Generating semantic descriptions from drawings of scenes with shadows. Technical report, MIT (1972)
9. Sugihara, K.: Mathematical structures of line drawings of polyhedrons. PAMI 4, 458–469 (1982)
10. Kanade, T.: A theory of origami world. Artificial Intelligence 13, 279–311 (1980)
11. Hoiem, D., Efros, A.A., Hébert, M.: Geometric context from a single image. In: ICCV, pp. 654–661 (2005)
12. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. PAMI 31, 824–840 (2009)
13. Barinova, O., Konushin, V., Yakubenko, A., Lee, K., Lim, H., Konushin, A.: Fast automatic single-view 3-d reconstruction of urban scenes. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 100–113. Springer, Heidelberg (2008)
14. Felzenszwalb, D., Veksler, O.: Tiered scene labeling with dynamic programming. In: CVPR (2010)
15. Criminisi, A.: Accurate visual metrology from single and multiple uncalibrated images. Springer, New York (2001)
16. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? PAMI (2002)