A Dynamic Programming Approach to Reconstructing Building Interiors

Anonymous ECCV submission

Paper ID 1389

Abstract. A number of recent papers have investigated the Manhattan world assumption, in which surfaces in the world are assumed to be aligned with one of three dominant directions [1–4]. 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 thousand manually annotated images, which are released in conjunction with this paper.

1 Introduction

In this paper we investigate the problem of reconstructing geometric models from single images. We choose to fit a model containing sparse geometric primitives with the intended aim of higher–level reasoning. If novel–view synthesis or photo-realistic 3D modelling were the goal then a dense reconstruction may be appropriate, but if the model is to be used as input to some reasoning problem, as is the motivation for the present work, then a simpler model may be more useful. In particular, when reasoning about indoor environments the location of the floor, ceiling, and wall planes provide strong cues to the probable locations of objects and events, whereas a raw point cloud would be more difficult to interpret. Furthermore, simple models have fewer degrees of freedom and are easier to infer from ambiguous image evidence.

It is perhaps for this reason that 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 investigated the even more constrained class of *indoor* Manhattan scenes [4, 5, 12], which consist entirely 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 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 [6] proposed a scheme to distinguish concave, convex, and occluding lines, which allows discrimination between possible and impossible objects. Waltz [7] investigated a more general problem involving incomplete line drawings and spurious measurements from shadows. Sugihara [8] proposed an algebraic approach to interpreting line drawings, while the “origami world” of Kanade [9] utilised heuristics to reconstruct a model of hollow shells and planar sheets. These approaches were limited to synthetic line drawings.

Hoiem *et al.* [10] and Saxena *et al.* [11] 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.* [12] 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], who have shown that line segments can be combined to generate indoor Manhattan models using a branch–and–bound algorithm. In contrast to their approach, 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 (see Figure 5), and obtain computation times several orders of magnitude less than their approach, as will be detailed in Section 7.

Furukawa *et al.* [3] have used the Manhattan world assumption for stereo reconstruction. Their models are of high quality but their quoted computation times (of between one minute and one hour forty minutes) make their approach unsuitable to on–line applications. Their approach is not comparable to ours because they use multiple calibrated views, and they search a different class of models.

3 Outline of Proposed Approach

The Manhattan world assumption states that world surfaces are oriented in one of three mutually orthogonal directions [1] and 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, and each corner is either concave, convex, or occluding (see 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 dominant directions. (Section 3.1)
2. Identify the floor and ceiling planes. (Section 3.2)
3. Warp the image to a canonical view. (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. We follow a similar approach to Kosreckà and Zhang [2], in which k–means clustering is used to obtain an initial estimate, after which EM is employed to identify the final vanishing points $\boldsymbol{v_1}$,$\boldsymbol{v_3}$, and $\boldsymbol{v_3}$.

If intrinsic camera parameters are not provided 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. For the remainder of this paper $\boldsymbol{v_1}$,$\boldsymbol{v_2}$, and $\boldsymbol{v_3}$ denote the calibrated vanishing points.

We assume that the vanishing point with largest absolute y–coordinate represents the up–down direction in the world (*i.e.* the direction in which gravity operates), and by convention we always label this direction $\boldsymbol{v_3}$. The horizon line is given by $\boldsymbol{h} = \boldsymbol{v_1} \times \boldsymbol{v_2}$.

3.2 Identifying the floor and ceiling planes.

An indoor Manhattan scene has exactly one floor and one ceiling plane, both with normal direction $\boldsymbol{v_3}$. 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 1). $H_{c\to f}$ is a planar homology with axis $\boldsymbol{h}$ and vertex $\boldsymbol{v_3}$ [13] and can be recovered given the image location of any pair of corresponding floor/ceiling points $(\boldsymbol{x_f}, \boldsymbol{x_c})$ as

$$H_{c\to f} = I + \mu \frac{\boldsymbol{v_3}\boldsymbol{h}^T}{\boldsymbol{v_3} \cdot \boldsymbol{h}} \ , \tag{1}$$

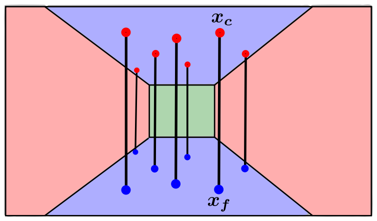

Fig. 1: The mapping $H_{c\to f}$ transfers points between the ceiling and floor.

where $\mu =< \boldsymbol{v_3}, \boldsymbol{x_c}, \boldsymbol{x_f}, \boldsymbol{x_c} \times \boldsymbol{x_f} \times \boldsymbol{h} >$ 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 $\boldsymbol{\hat{x}_c}$ from the region above the horizon in the Canny edge map, then we sample a second point $\boldsymbol{\hat{x}_f}$ collinear with the first and $\boldsymbol{v_3}$ 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 Warping the image to a canonical view

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 a homography

$$H = \begin{pmatrix} \boldsymbol{v_3} \times \boldsymbol{e_3} \\ \boldsymbol{v_3} \\ \boldsymbol{v_3} \times \boldsymbol{e_3} \times \boldsymbol{v_3} \end{pmatrix} , \tag{2}$$

where $\boldsymbol{e_3} = [0, 0, 1]^T$.

This step is not crucial to our approach and we could instead do all geometric calculations in the original image. Hence, the fact that the warped image will be contain a singularity if $\boldsymbol{v_3}$ is within the image bounds does not restrict the applicability of our approach. For the sake of simplicity, however, we present the remainder of this paper assuming that $\boldsymbol{v_3}$ is outside the image bounds.

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, 10, 11].

We adopt the simple and efficient line–sweep approach of Lee *et al.* [4], which assigns one of four labels $\{1, 2, 3, z\}$ to each pixel, where z represents an unknown orientation and the remaining labels identify the normal direction of the surface that generated the pixel. We denote the orientation map $o : \mathbb{R}^2 \rightarrow \{1, 2, 3, z\}$.

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 [10] or Saxena [11], could be used instead.

We generate four binary images B_i such that pixel $B_i(\boldsymbol{x}) = 1$ if and only if $o(\boldsymbol{x}) = i$. We then compute the integral image [14] for each B_i, which allows us to count the number of pixels of a given orientation within any rectangular sub–image in $O(1)$ time. This represnetation expediates evaluation of the cost function described in the following section.

4 Problem statement

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 $a_i \in \{1, 2\}$, which determines whether its vanishing point is $\boldsymbol{v_1}$ or $\boldsymbol{v_2}$, and a row index r_i at which its upper edge meets the corner to its left. An example model is shown in Figure 2b. Remarkably, this simple description completely specifies all four corners of each wall segment (see Figure 2a). Clockwise from top–left the vertices of the i$^{\text{th}}$ wall are

$$\boldsymbol{p_i} = [c_i, r_i, 1]^T \tag{3}$$
$$\boldsymbol{q_i} = \boldsymbol{p_i} \times \boldsymbol{v_{a_i}} \times [1, 0, -c_{i+1}]^T \tag{4}$$
$$\boldsymbol{r_i} = H_{c \rightarrow f} \boldsymbol{q_i} \tag{5}$$
$$\boldsymbol{s_i} = H_{c \rightarrow f} \boldsymbol{p_i} \ . \tag{6}$$

A model $\mathcal{M}$ predicts one of three possible surface orientations $\mathcal{L} = \{1, 2, 3\}$ for each pixel $\boldsymbol{x}$. We denote this prediction $\mathcal{M}(\boldsymbol{x}) \in \mathcal{L}$ where

$$\mathcal{M}(\boldsymbol{x}) = \begin{cases} a_i & \text{if } \boldsymbol{x} \in \text{quad}(\boldsymbol{p_i}, \boldsymbol{q_i}, \boldsymbol{r_i}, \boldsymbol{s_i}) \\ a_v & \text{otherwise} \end{cases} , \tag{7}$$

where a_v is the index of the vanishing point representing the vertical direction in the world.

Not all models $\mathcal{M}$ are physically realisable, but those that are not are easily 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; we simply note 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.

We are now ready to state the minimisation problem formally. Given an input image of size $W \times H$ and an orientation map o, we define the cost of assigning

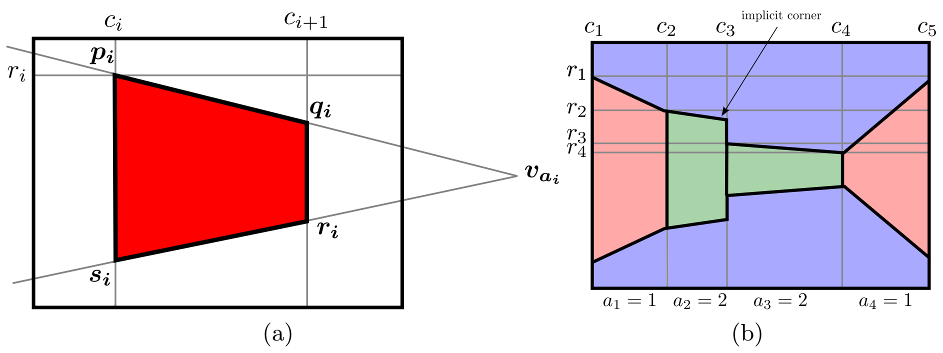

Fig. 2: (a) An illustration of the model $\mathcal{M} = \{c_1, (r_1, a_1), ..., (r_4, a_4), c_5\}$; (b) the values (c_i, r_i, a_i, c_{i+1}) fully specify the four corners of a wall.

pixel $\boldsymbol{x}$ label $l \in \mathcal{L}$ as

$$C_p(\boldsymbol{x}, l) = \begin{cases} 0, & \text{if } o(\boldsymbol{x}) = l \text{ or } o(\boldsymbol{x}) = z \\ 1, & \text{otherwise} \end{cases} . \tag{8}$$

The overall cost for a model $\mathcal{M}$ is then

$$C(\mathcal{M}) = \sum_{\boldsymbol{x} \in \mathcal{I}} C_p(\boldsymbol{x}, \mathcal{M}(\boldsymbol{x})) \tag{9}$$

and we seek the model $\mathcal{M}^*$ with minimum cost

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmax}}\ C(\mathcal{M}) . \tag{10}$$

Finally, the marginal cost C_w of the i^{th} wall $W_i = (r_i, a_i)$ is

$$C_w = \sum_{x_i \leq x \leq x_{i+1}} \Big(\sum_{1 \leq y \leq U_i} C_p(x, y, a) + \sum_{U < y \leq L_i} C_p(x, y, a_v) + \sum_{L < y \leq H} C_p(x, y, a) \Big) \tag{11}$$

where L_i and U_i are respectively the ceiling/wall and floor/wall intersection at column x_i. The three sums over y are $O(1)$ operations given the integral images described in Section 3.4.

5 Proposed algorithm

In this section we present a dynamic programming solution to the problem posed in Section 4. We develop the algorithm incrementally to assist readability. Our solution rests on a decomposition of the indoor Manhattan problem into a set of sub–problems that can be solved in terms of one another. Consider the following sub–problem.

Let $f_{in}(x, y, a, k)$ be the cost of a model $\mathcal{M}^+ = \{c_1, W_1, ..., W_{n-1}, c_n\}$ such that

1. $n = k$, (*i.e.* $\mathcal{M}^+$ contains k corners),
2. $c_n = x$ (*i.e.* the rightmost corner is at column x),
3. $W_{n-1} = (y, a)$ (*i.e.* wall $n-1$ terminates at row y with orientation a),
4. $\mathcal{M}^+$ is feasible, and
5. $\mathcal{M}^+$ has minimal cost among all such models.

We show in the additional material that if a model

$$\mathcal{M} = \{c_1, W_1, ..., W_{k-1}, c_k\} \tag{12}$$

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

$$\mathcal{M}' = \{c_1, W_1, ..., W_{k-2}, c_{k-1}\} \tag{13}$$

is a solution to the sub–problem $f_{in}(c_{k-1}, r_{k-1}, a_{k-1}, k-1)$.

In light of this fact we introduce the following recurrence relation:

$$f_{in}(x, y, a, k) = \min_{x'<x,\ y',a'} \Big(f_{in}(x', y', a', k) + C_w \Big) \ , \tag{14}$$

where C_w is the marginal cost of the wall $W = (y', a')$. We also introduce the following boundary conditions:

$$f_{in}(x, y, a, k) = \begin{cases} 0, & \text{if } x = 0 \\ \infty, & \text{if } k < 0 \text{ and } x > 0 \end{cases} \ , \tag{15}$$

which are justified, respectively, because a model with rightmost corner at $x = 0$ does not span any part of the image, so its cost must be zero, and because a model cannot contain less than zero corners.

Finally, the cost of the optimal model (10) is

$$C(\mathcal{M}^*) = \min_{\substack{1 \le y \le H \\ k \le K \\ a \in \{1,2\}}} \Big(f_{in}(W, y, a, k) - \lambda k \Big) \ . \tag{16}$$

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

We compute $C(\mathcal{M}^*)$ by recursively evaluating f_{in} according to (14) until we reach one of the boundary conditions (15). In line with the standard dynamic programming approach we cache each evaluation to avoid evaluating the same node multiple times. Once all nodes have been evaluated, $\mathcal{M}^*$ is obtained by back–tracking through the evaluation graph.

Complexity. Since each node is evaluated at most once, the overall complexity is given by the product of the number of nodes and the complexity of evaluating each node. The latter is $O(W^2H)$, since the minimisation in (14) is over $O(WH)$ terms and computing the marginal cost C_w requires $O(W)$ additions. The number of nodes is $O(WHK)$, so the overall complexity of the basic algorithm is $O(W^3H^2K) = O(L^5K)$ where $L = \max(W, H)$.

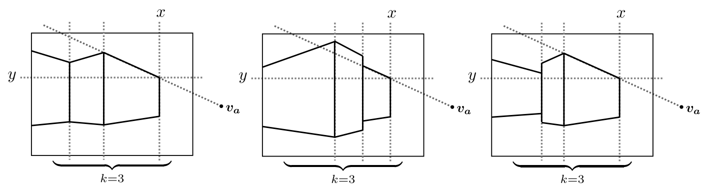

Fig. 3: Three models that satisfy constraints 1–4 for the sub–problem $f_{in}(x, y, a, k)$. There will only be one such model that satisfies the minimal cost constraint. Note that the right–most wall in each model terminates at (x, y) with orientation a.

5.1 Auxiliary nodes

The basic algorithm is hampered by the need to minimise simultaneously over x' and y' in (14). In this section we show how to decouple these minimisations by introducing auxiliary sub–problems, which results in a significant reduction in computational complexity.

We introduce three new sub–problems: f_{up}, f_{down}, and f_{out}. Each of these is defined identically to f_{in}, except that constraint 3 is replaced according to the following table:

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

Using a similar argument to that given in the previous section it can be shown that these sub–problems are related by the following recurrence relations:

$$f_{out}(x, y, a, k) = \min_{a' \in \{1,2\}} \min \begin{cases} f_{up}(x, y, a', k) \\ f_{out}(x, y, a', k) \\ f_{down}(x, y, a', k) \end{cases} \tag{17}$$

$$f_{up}(x, y, a, k) = \begin{cases} \min\Big(f_{in}(\cdot), f_{up}(x, y-1, a, k)\Big), & \text{if } y \geq 1 \\ \infty, & \text{otherwise} \end{cases} \tag{18}$$

$$f_{down}(x, y, a, k) = \begin{cases} \min\Big(f_{in}(\cdot), f_{down}(x, y+1, a, k)\Big), & \text{if } y \leq H \\ \infty, & \text{otherwise} \end{cases} \tag{19}$$

$$f_{in}(x, y, a, k) = \begin{cases} 0, & \text{if } x = 0 \\ \infty, & \text{if } k < 0 \\ \min_{x' < x}\Big(f_{out}(x', v(x'), a, k-1) + c(x')\Big), & \text{otherwise} \end{cases} \tag{20}$$

where in the final equation $v(x')$ is the y–coordinate at which the line through $\boldsymbol{v_a}$ and (x, y) meets column x'. Feasibility is enforced by removing either or both

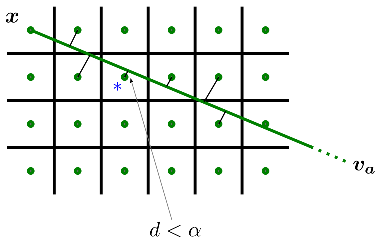

Fig. 4: A line from $\boldsymbol{x}$ to $\boldsymbol{v_a}$, and the distances d to nearby pixel centres (green dots). The starred pixel is the first that satisfies $d < \alpha$.

of the f_{up} or f_{down} terms in (17) if such a corner would result in an infeasible model.

5.2 The line–jump optimisation

Evaluating a f_{in} node remains an $O(W)$ operation due to the minimisation over x' in (20). In this section we show how to reduce this to an $O(1)$ operation.

Consider Figure 4, showing a line from a pixel at $\boldsymbol{x}$ to a vanishing point. If the line passed exactly through some other pixel centre $\boldsymbol{x_p}$ then we could simply evaluate $f_{in}(x_p, y_p, a, k)$ and terminate the line search at that point. It is unlikely that the line will pass exactly through an integer–valued pixel centre, so we instead specify a threshold α within which a pixel centre will be considered to fall exactly on a line. We suffer no loss of precision by doing this since the image itself is precise only up to pixelation. We now have

$$f_{in}(x,y,a,k) = \min \begin{cases} \min_{x_p \le x' < x} \Big(f_{out}(x', v(x'), a, k-1) + c(x') \Big) \\ f_{in}(x_p, v(x_p), a, k-1) + c(x_p) \end{cases} . \tag{21}$$

where (x_p, y_p) is the closest pixel to (x, y) satisfying $x_p < x$ and for which $d = \boldsymbol{x} \cdot (\boldsymbol{x_p} \times \boldsymbol{v_a})$ is less than α. Both (x, y) and $\boldsymbol{v_a}$ have integer coordinates since they are observed at pixel locations, so we have the upper bound

$$x - x_p < \beta \tag{22}$$

for some constant β, which is dependent only on our choice for α [15].

Complexity. Evaluating each node is now an $O(1)$ operation, so the overall complexity of our algorithm is given by the number of nodes in the evaluation graph, which is

$$O(KL^2) \; . \tag{23}$$

5.3 Down–sampling the orientation map

As a further optimisation we down–sample the orientation map o, replacing each $m \times m$ block of cells with a single cell containing the majority vote over the block. This reduces both the number of nodes in the graph and the cost of evaluating C_w.

6 Other approaches

Graph cuts. Many pixel–labelling problems have been successfully solved using graph cuts. Kolmogorov and Zabih [16] have investigated the class of cost functions that can be minimised using graph cuts. Their work demonstrates that *regularity*, a special case of sub–modularity, is a necessary condition for any energy function to be minimised via graph cuts.

It turns out that the cost (9) is not regular, so cannot be minimised using graph cuts. Intuitively this is because the indoor Manhattan constraint induces complicated dependencies between the pixels in each column. [1] Furthermore, applying graph cuts to this problem would entail using a technique such as α–expansion [16], which is both approximate and non–deterministic. In contrast, our approach is exact and deterministic.

Branch and bound. Lee *et al.* [4] proposed a branch–and–bound solution to the indoor Manhattan problem. Their approach identifies straight lines in the image and then combines them in various permutations to generate a set of model hypotheses, each of which are evaluated using a cost function similar to (9). Whereas their search space grows combinatorially with the number of lines in the model, our approach is linear in model complexity and hence we can model significantly more complex scenes. We give timing comparisons with their approach in Figure 7.

Their approach also differs from ours in that they only allow wall boundaries to occur where lines are observed in the image. Our system can easily be extended to enforce such a constraint by removing f_{in} terms from (18) and (19) for locations where lines are not detected. However, we have found this to be unnecessary because the cost (9) implicitly favours models with corners at the location of observed lines since they will fit the orientation estimate o better. Furthermore, avoiding an explicit dependency on a line detector is favourable because our system is not reliant on the successful detection of structurally important line. In contrast, Lee *et al.* are unable to build the correct model if many structurally important edge are undetected. We have found that the structurally important edges are often the least salient, since walls in a room are often painted the same colour, so image gradients at their boundaries are generated only by subtle lighting differences arising from the different surface orientations.

7 Results

We tested our system on a dataset of 634 manually annotated images of indoor scenes. To expedite the annotation process 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.

[1] For example, if some pixel $\boldsymbol{p}$ is assigned label a then $\boldsymbol{q} = H_{c\rightarrow f}\boldsymbol{p}$ must be assigned the same label, even though the two may be arbitrarily far from one another in the image.

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 $\alpha = 0.01$, $K = 7$, $m = 4$, $\lambda = 100$. Image sizes were 640×480 pixels. We found our approach to be very 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 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.

7.1 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, 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.

8 Conclusion

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. Future work will investigate the use of richer cues for obtaining initial orientation estimates since our results indicate that is currently the limiting factor of our approach.

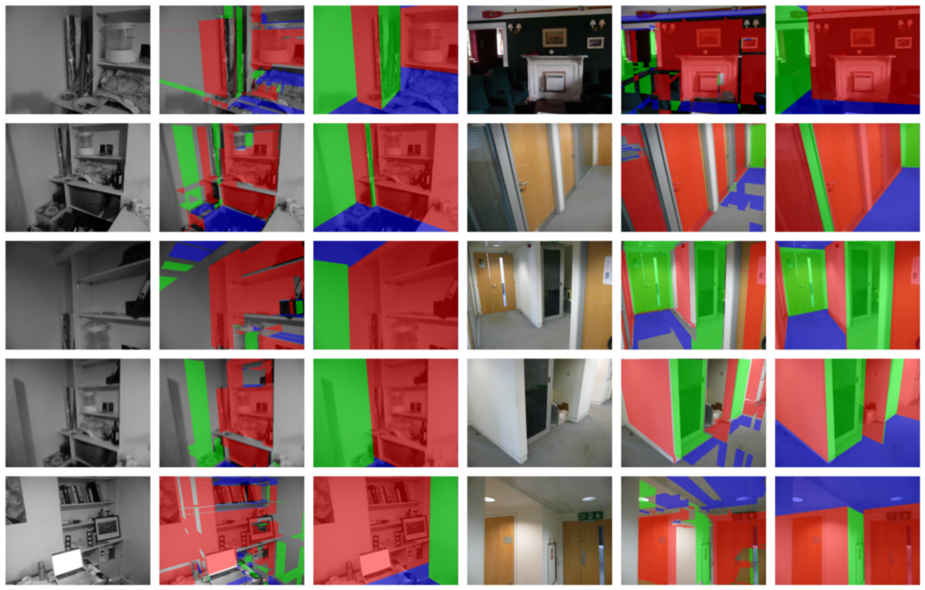

Fig. 5: Models estimated by our algorithm. Each panel contains three images: the original image, the initial orientation estimate, and the final model output by our system. Best viewed in colour.

References

1. Coughlan, J., Yuille, A.: Manhattan world: compass direction from a single image by bayesian inference. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Volume 2. (1999) 941–947 vol.2
2. Koseckà, J., Zhang, W.: Video compass. In: Proc 7th European Conf on Computer Vision. Volume 2353 of Lecture Notes in Computer Science., Springer (2002) 4: 476–490
3. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Manhattan-world stereo. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on **0** (2009) 1422–1429

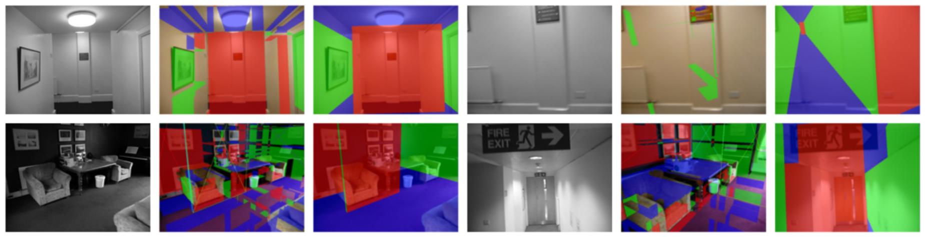

Fig. 6: Failure cases of our system. Best viewed in colour.

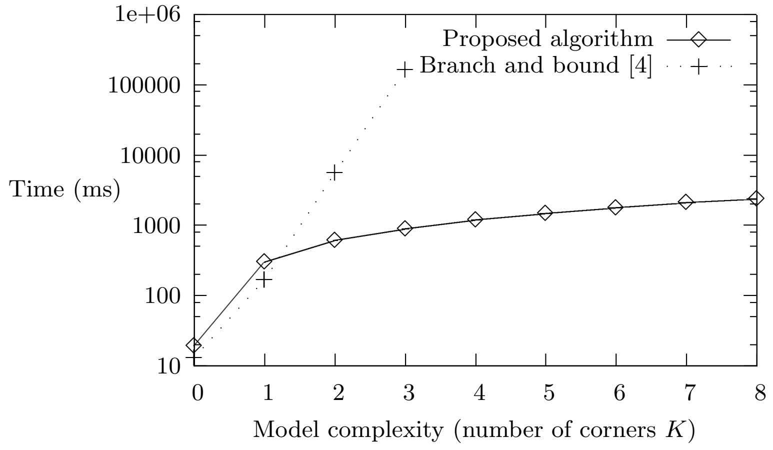

Fig. 7: Efficiency comparison with the branch–and–bound of Lee *et al.* [4] (our own implementation). Our approach was able to fit highly complex models in a under 2 seconds. Their algorithm took more than three minutes to fit a model of 3 corners; most of this time was spent evaluating the cost of the 200,000 building hypotheses.

4. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). (2009)
5. Flint, A., Mei, C., Reid, I., Murray, D.: Growing semantically meaningful models for visual slam. In: Proc 28th IEEE Conf on Computer Vision and Pattern Recognition. (2010)
6. Huffman, D.A.: Impossible objects as nonsense sentences. Machine Intelligence **6** (1971) 295–323
7. Waltz, D.L.: Generating semantic descriptions from drawings of scenes with shadows. Technical report, Cambridge, MA, USA (1972)
8. Sugihara, K.: Mathematical structures of line drawings of polyhedrons-toward man-machine communication by means of line drawings. Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-4** (1982) 458 –469
9. Kanade, T.: A theory of origami world. Artificial Intelligence **13** (1980) 279–311
10. Hoiem, D., Efros, A.A., Hébert, M.: Geometric context from a single image. In: Proc 10th IEEE Int Conf on Computer Vision. (2005) 654–661
11. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009) 824–840
12. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: Proc 12th IEEE Int Conf on Computer Vision. Volume 2. (2009)
13. Criminisi, A.: Accurate visual metrology from single and multiple uncalibrated images. Springer-Verlag New York, Inc., New York, NY, USA (2001)
14. Viola, P.A., Jones, M.J.: Robust real-time object detection. International Journal of Computer Vision **57** (2004) 137–154
15. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons (1998)

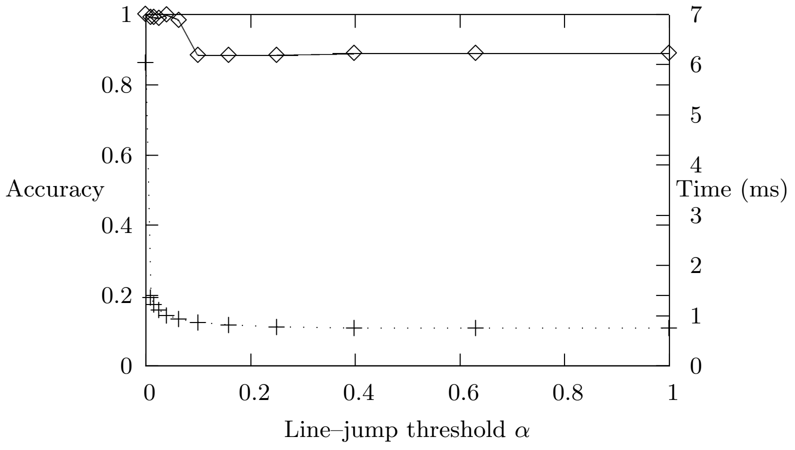

Fig. 8: The line–jump parameter α trades off accuracy (crosses) for efficiency (diamonds). Accuracy is computed relative to the baseline achieved when $\alpha = 0$. The first few increments of α exhaust most of the efficiency gains, and at this end of the graph the loss in accuracy is negligible. Based on these results we set $\alpha = 0.01$ in our remaining experiments.

16. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence (2002)