Recognising planes in a single image

Osian Haines and Andrew Calway

Abstract—We present a novel method to recognise planar structures in a single image and estimate their 3D orientation. This is done by exploiting the relationship between image appearance and 3D structure, using machine learning methods with supervised training data. As such, the method does not require specific features or use geometric cues, such as vanishing points. We employ general feature representations based on spatiograms of gradients and colour, coupled with relevance vector machines for classification and regression. We first show that using hand-labelled training data, we are able to classify pre-segmented regions as being planar or not, and estimate their 3D orientation. We then incorporate the method into a segmentation algorithm to detect multiple planar structures from a previously unseen image.

Index Terms—Planar structure, single images, recognition, learning.

1 Introduction

This paper is concerned with the automatic extraction of 3D structure from single images. While the creation of 3D models of real-world scenes from image data has been a topic of interest for a long time, it is usual for this to involve either multiple views of a scene or video data, exploiting parallax to obtain information about scene depth. Inferring depth from only a single image is much more challenging. However, previous works have shown that a number of image cues can be exploited to extract information about depth, shape, or other 3D structure. For example, vanishing points can be used to calculate plane orientation [22] [27], estimate distances [7], measure lengths [12] and construct intricate line models [31]. Shape from shading techniques allow estimation of surface normals [30], whilst shape from texture methods exploit distortions in appearance to give estimates of surface shape [10].

But such methods tend to be rather restrictive in terms of the images they can deal with, e.g. requiring 'Manhattan'-like environments with three dominant orientations in the case of vanishing points [22]. More recent approaches have therefore explored the potential for making use of the relationship between image appearance and structure, learned from examples. Such methods, based on machine learning techniques, use information gleaned from a training set to predict structure in new images, avoiding the need to impose models on the data. The method described here falls into this class.

Two prominent existing methods are those of Saxena et al. [32] and Hoiem et al. [19]. The former is able to recover an approximate depth map for an image having learned the relationship between image appearance and ground truth depth maps derived from laser scanning; this can be used for creating virtual fly-throughs in generated 3D models, or for reconstruction from widely separated image sets [32]. In contrast, Hoiem et al.'s work focuses on segmenting an image into areas representing coarse geometric classes based on labelled image data. It distinguishes horizontal and vertical surfaces from non planar areas such as the sky, with the vertical surfaces being further classified into either forward, left or right facing. The method can be used to generate 'pop-up' 3D models, or as a prior for object detection [18].

• *Bristol.*
E-mail: haines/andrew @cs.bris.ac.uk

1.1 Contribution

The main contribution here is that we show that as well as detecting planar structure it is also possible to estimate 3D orientation using a learning based method. This is different from the approaches above, which are either not able to identify actual planar structures [32], or do not provide an accurate estimate of 3D orientation [19]. It therefore sits between them – detecting planar structure and estimating relative depth via 3D orientation – defined within a common framework. Our motivation is the number of applications which would benefit from early detection and accurate orientation estimation of planar structure, notably in real-time 3D reconstruction applications such as simultaneous localisation and mapping (SLAM), see e.g. [26] [16]. As such, we make the assumption that we have access to the camera calibration parameters during training, and that these remain constant for all train and test data: it is this consistency which allows us to predict plane orientation vectors, rather than coarse classifications. Otherwise, our approach uses only the cues learned from a labelled training set, and does not require any other pre-specified priors.

We adopt techniques from machine learning and image segmentation, including a relevance vector machine classifier and a Markov random field (MRF) segmentation algorithm, but combine them in a novel

way. As well as being motivated by the success of the methods noted above, the approach is also inspired by theories of human vision, which suggest that humans are able to recognise complex structures, even when stereo or parallax cues are unavailable, by virtue of their prior experience with the world [11], and a form of top-down interpretation guided by what they expect to see [29].

We first describe an algorithm to learn the relationship between appearance and planar structure from examples, which enables the recognition of planes in individual image regions and the estimation of their orientation. This is done using features built from gradient and colour descriptors, within a bag of visual words model, and we use relevance vector machines for classifying planar structure and regressing orientation. This works on individual, pre-segmented image regions (so assumes a relevant region of interest has been defined), giving for each a classification (whether it is planar or not) and an orientation, expressed as a normal vector with respect to the camera: we refer to this as plane recognition. For this baseline recognition algorithm we report classification accuracy of around 92% and a median orientation error of 10°, when tested on a test set of image regions gathered independently of the training data, from a separate urban location.

Next we use plane recognition within a graph-based segmentation to detect multiple planes from an image without prior knowledge of their locations. In contrast to plane recognition, this stage is able to find planes and their extent from the entire image, then assign them each a class and estimate their orientation: we call this plane detection (see Fig. 1 for examples). We present two versions of this algorithm: a version which uses saliency detection to focus on regions of the image with interesting texture [14], and a second which uses a regular grid of points, to enable plane detection using all parts of an image.

The first method is shown to be in general more accurate, while the second achieves greater coverage over the scene. Results on data from urban environments demonstrate that these techniques are effective, giving an 83.6% classification accuracy and median orientation error of 16.2° when using salient points, or an accuracy of 81.6% and orientation error of 16.3° when using the grid-based method (again on independent test data). We believe this constitutes good performance given the difficulty of the task, especially considering this is something not achieved before by other methods – namely recovering an accurate orientation for large-scale planar structures.

In the next section we review previous work. An overview of our method is given in Section 3, followed in Sections 4 and 5 by details of the recognition algorithm and segmentation frameworks respectively. Results of experiments are presented in Section 6, including a comparison with recent work. Earlier versions of the work were previously reported in [15] and [14]; this work brings the two together for a more coherent exposition, adding results of new experiments and analysis, and an alternative version of the method. Further in-depth analysis can be found in [13].

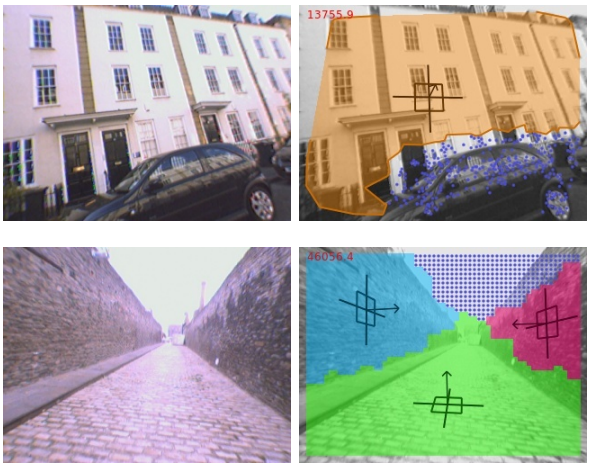


Fig. 1. Example results of our algorithm. From images such as those on the left, we can detect planes and predict their 3D orientation, as shown on the right. The two versions of our method segment using either salient points (top) or a regular grid of points (bottom).

2 Related Work

Here we discuss examples of prior work on extracting planar structure from single images. This can be divided into two main categories: methods which explicitly use geometric properties, such as parallel lines and texture; and work which aims to recognise structure based on learning from training examples.

Given two or more sets of parallel lines lying on a plane, their respective vanishing points uniquely define the plane's 3D orientation [17]. Thus, detecting such line features in an image enables the extraction of structure, providing they lie on a common plane. One approach is to detect rectangular structures, such as windows or doors, which provide orthogonal parallel lines in the same plane [22] [27], which can be used for basic camera pose recovery and wide baseline matching. However, these methods rely on orthogonal, Manhattan-like structure, and reliable line detection, hence limiting applicability.

In shape from texture methods, the deformation of imaged texture is related to surface shape, to recover orientation or curvature properties. Work by Gårding [10], for example, uses the visible compression of texture under projection to determine slant and tilt for planar surfaces. In such methods as these it is notable that detection is generally not addressed – it is assumed that the image contains a single planar surface.

An interesting crossover between the geometric methods above, and those that use machine learning,

is the work of Barinova et al. [1], who use estimation of the location of a scene's principal vertical structures and the horizon line, together with a classification of line segments to identify horizontal lines. This method is also able to give an approximate 3D reconstruction of city scenes, but does not go so far as producing precise orientation estimates for the resulting planes.

2.1 Machine Learning Methods

More recent work has looked at techniques for learning the relationship between appearance and structure. A good example of such a technique is by Torralba and Oliva [35], who estimate overall depth using knowledge that certain types of structure tend to appear at particular distances. However, this work focuses on global scene properties, which is a level of understanding too coarse for most interesting applications.

Saxena et al. [32] go further than this by estimating whole-image depth maps based on training images labelled with absolute depth. They can create basic 3D models of the scene, built from locally planar facets, and their comparison to ground truth depth shows good accuracy. However, the resulting models do not explicitly represent higher-level structures – The superpixel segments are all assumed to be locally planar, and accuracy of planar facet orientations is not reported. Rather, the focus of the work is to produce visually plausible renderings of the scene, which are assessed by human subjects. This is in contrast to the work we present here, where we explicitly aim to find large-scale planes in the image, and to assign them an accurate orientation.

Hoiem et al. [19] describe an approach most similar to ours, where a variety of texture and colour features, as well as explicit vanishing point information, are used to classify planar segments into coarse geometric classes, distinguishing between support surfaces (horizontal), and left, right, or front facing vertical surfaces. This coarse classification is used to produce a general scene layout, creating simple 'pop-up' 3D models, or as a prior for object recognition.

Their classification of the image into geometric classes bridges the gap between semantic understanding and 3D reconstruction. However, because orientations are coarsely quantised it means that the recovered 3D models lack specificity, being unable to distinguish similarly oriented planes. Moreover, their requirement that the camera be roughly aligned with the ground plane, and the use of vanishing point information as a cue, suggest they are not making use of fully general information. What we are aiming for, on the other hand, is a more general treatment of prior information. The main difference to our presented method is that they do not offer an estimate of actual plane orientation, as we do, but quantise the results into to a small number of discrete geometric classes.

3 Overview

Our method consists of two main components: a plane recognition algorithm; and a graph-based detection stage. The former takes image regions, classifies them as either planar or non-planar, and for planes provides an estimate of their 3D orientation in the camera coordinate frame. This forms the core element within the detection stage, which identifies multiple planes in an image from localised plane recognition results. Figures 2 and 4 illustrate the key elements in each component. A summary of each is given below and full details are given in Sections 4 and 5.

The plane recognition algorithm has three components [15]: appearance representation; planar classification; and orientation regression. To encode appearance we use a representation which characterises the distribution of local patterns centred around a set of points. We use a combination of histograms of gradients and colour for this, quantised using a bag of words codebook; followed by dimensionality reduction using a variant of latent semantic analysis, to give sets of latent 'topics'. The spatial distribution of the topics over the salient points is then captured using spatiograms, which are used to train a classifier to predict whether regions are planar or not, and a regression algorithm to predict plane normals.

To detect planes in an image [14], we apply the plane recognition algorithm to multiple overlapping image regions, giving plane/non-plane classifications and orientation estimates for each. Points are sampled from across the image and given estimates of planarity and orientation based upon the regions in which they lie. This can be used to separate planar regions from non-planar; we use mean shift clustering on the set of normals in the image to obtain the dominant orientations, and use this to separate distinct planes. Segmentation of individual planes is then achieved in a graph-based algorithm, using iterative conditional modes, which also allows a degree of localised smoothing. Plane recognition is then reapplied to each planar region to obtain an updated plane orientation estimate. The result is a set of planar segments, plus points otherwise belonging to non-planar regions, such as those shown in Fig. 1. We show results using both a sparse set of salient points, and a version using a multi-scale grid of points.

4 Plane Recognition

In this section we describe the plane recognition algorithm, which classifies image regions as being planar or not, and for the former provides a 3D orientation estimate. We emphasise that this works on individual, pre-segmented image regions only, and does not apply to the image as a whole; marking the relevant planar or non-planar image region is part of the data acquisition process. The main components are shown in Fig. 2.

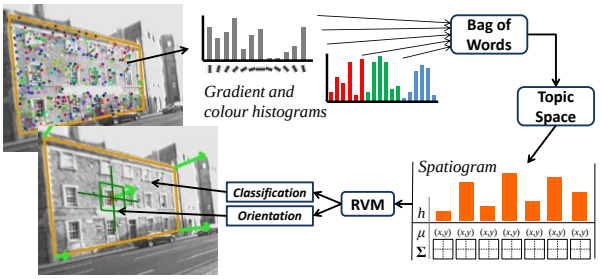


Fig. 2. Main components of the plane recognition algorithm.

4.1 Training Set

We gathered a training set of regions containing planar surfaces and their orientations, plus examples of non-planar regions, selected from video sequences of urban locations. We mark up regions of interest, labelling them as planar or non-planar as appropriate. For the planar regions a ground truth orientation is defined using an interactive method based on vanishing points. Four points corresponding to the corners of a rectangle lying on the plane in 3D are marked up by hand and the pairs of opposing edges are extended until they meet to give vanishing points in two orthogonal directions. Joining these gives the vanishing line $\mathbf{l} = \mathbf{v_1} \times \mathbf{v_2}$ of the plane, where $\mathbf{v_1}$ and $\mathbf{v_2}$ are the vanishing points in homogeneous coordinates.

The plane normal can then be obtained from $\mathbf{n} = \mathbf{K}^{\mathrm{T}}\mathbf{l}$, where $\mathbf{K}$ is the 3×3 intrinsic camera calibration matrix [17]. It is this relationship between camera parameters and plane orientation that makes it possible to recover accurate plane orientations for new data, because the relationship between image appearance and orientation is constant. This requires a consistent and known camera calibration for all images used, which is consistent with our intended application area of SLAM and 3D reconstruction, but makes running our algorithm on other existing datasets problematic.

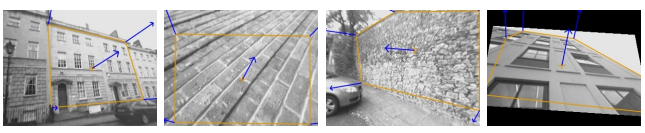


Fig. 3. Examples of hand-segmented regions and their ground truth orientation (rightmost image is obtained by warping).

To further increase the size of training set, we synthetically generate new variations from the marked-up set, first by reflecting all the regions about the vertical axis; then we generate examples of planes with different orientations by warping the regions – effectively simulating the view as seen by a camera in different poses [17]. Examples of training data are shown in Fig. 3.

4.2 Salient Points

An image region will generally contain a large amount of visual information, as well as potentially less informative blank regions. To create a more compact representation, and focus on parts of the image which are more likely to be useful, we select a subset of salient points in the image around which to concentrate further processing. This is achieved with the difference of Gaussians (DoG) saliency detector [24], which selects blob-like regions in the image. This gives a scale as well as a location for each point, which we found to be beneficial [15]. Note that the image descriptors are built from the patches around the detected points, not just at the points themselves, i.e. we are doing more than labelling individual points. An alternative method of sampling the image is also discussed in Section 5.3.

4.3 Image Descriptors

Image descriptors are created in the region about each salient point, where the region size is dictated by the scale returned by the saliency operator. We use two complementary feature descriptors: the first is gradient orientation histograms to describe texture, which consist of histograms of edge orientation, computed by applying edge filters to the image. We create four histograms per patch, one for each quadrant, comprised of 12 angular bins covering the range $[0, \pi)$, and concatenated to give a 48D descriptor.

Secondly, we represent colour using RGB histograms, created by concatenating intensity histograms from the red, green and blue channels of the patch. Each has 20 bins, giving a 60D descriptor. The importance of colour for classifying structure was demonstrated by [19], and as we hoped, combining both types of descriptor gives superior performance to either in isolation (see Section 6.1.1). However, colour is not beneficial for estimating orientation, and so we maintain separate representations for classification and regression, the former comprising gradient and colour information, the latter with gradient only.

4.4 Bag of Words

To further reduce dimensionality, we represent the distribution of descriptors in a region using a bag of words approach [25]. We identify clusters in descriptor space and their centres then form 'visual words', creating a 'vocabulary' codebook. Two separate vocabularies are created, to represent the gradient and colour descriptor spaces – created by running K-means on a set of 100 representative images. Regions are then compactly represented by a pair of word histograms, expressing the occurrence of gradient and colour words from the respective vocabularies.

Each word histogram, stored in a vector $\mathbf{h}^{\mathsf{word}}$, has K elements $\{h_k^{\mathsf{word}} | k = 1 \dots K\}$, where K is the size

of the vocabulary, and these are found by quantising each of the descriptors at the salient points to the closest word, i.e. $h_k^{\text{word}} = |\Lambda_k|$, where Λ_k is the set of points whose descriptors have been quantised to cluster (and hence word) k. We also apply term frequency – inverse document frequency weighting [25], to weight words according to their relative importance in the corpus.

4.5 Topic Discovery

A problem with the standard bag of words model is that there is no association between words, and no way to account for the fact that multiple words may correspond to the same semantic concept – any two non-identical words are considered the same distance from each other. Furthermore, as vocabulary size increases, in order to represent richer environments, word histograms will become very high dimensional and sparse, making meaningful comparisons difficult.

A solution to this is to use a latent topic model, where a set of salient 'topics' amongst the words in a corpus are extracted; this can be considered a dimensionality reduction technique, resulting in vectors of topic occurrence, instead of word histograms. We use orthogonal non-negative matrix factorisation (ONMF) [5] for this. ONMF creates a linear model, similar to that of latent semantic analysis (LSA) [8], and has the property that new topic vectors can be obtained simply by matrix projection of word histograms. It also ensures that all projected topic coefficients are non-negative (not just the original factor matrices), which is essential for creating spatiograms (see next section). Furthermore, ONMF does not require the rather complicated variational methods or Gibbs sampling necessary for latent Dirichlet allocation [4].

ONMF factorisation is applied to the training set, by factorising the $K \times N$ term-document matrix $\mathbf{H}^{\text{word}}$ (whose columns correspond to the word histograms for the N training set regions) as $\mathbf{H}^{\text{word}} \approx \mathbf{W}\mathbf{H}^{\text{topic}}$, where the columns of $\mathbf{W}$ ($K \times T$) are the basis of the latent topic space (of rank T, the number of topics), and $\mathbf{H}^{\text{topic}}$ ($T \times N$) contains the topic histograms for each region. The word histogram for region n can then be approximated by $\mathbf{h}_n^{\text{word}} \approx \mathbf{W}\mathbf{h}_n^{\text{topic}}$, where $\mathbf{h}_n^{\text{topic}}$ is the nth column of $\mathbf{H}^{\text{topic}}$. Since $\mathbf{W}$ is constrained to be orthogonal (in contrast to standard non-negative matrix factorisation), a topic histogram for a region can therefore be calculated by projecting its word histogram into the topic space, i.e. $\mathbf{h}^{\text{topic}} = \mathbf{W}^{\mathrm{T}}\mathbf{h}^{\text{word}}$.

A further benefit of this topic analysis is that it allows the combination of information from the two feature spaces into a single topic space. This can be done by concatenating the gradient and colour term-document matrices before running ONMF. As discussed earlier, we use both types of feature for classifying planes but only gradient features for estimating orientation, so it is necessary to maintain two topic spaces, for classification and regression. In terms of implementation, ONMF factorisation has no closed form solution and so we use the iterative method described in [5].

4.6 Spatiograms

Word and topic histograms compactly represent regions so that they may be classified; however, we found performance to be somewhat disappointing (c.f. Section 6.1.1). A disadvantage of the standard bag of words model is that all location information is discarded – whereas the relative position of features is likely to be important for characterising planar structure. While it is possible to include spatial information using constellation or star models [9], these are computationally expensive, and scale poorly to large numbers of words. Another alternative would be spatial pyramid models, which build a representation over multiple locations and scales, but these are better suited to describing whole images rather than regions, and require concatenation of descriptors over many pyramid levels [23].

Instead we use spatiograms as proposed in [3]. A spatiogram is a generalisation of a histogram, where each bin contains not only a count but also the mean and covariance of spatial positions of points which contributed to it. Spatiograms have been used successfully for object detection and tracking [28] using colour and gradient histograms (but not, as far as we are aware, using bags of words). We use them here since they are a relatively compact way of representing the spatial distribution of words or topics.

Given a word histogram $\mathbf{h}^{\text{word}}$, its corresponding spatiogram $\mathbf{s}^{\text{word}}$ is comprised of a set of K triplets $\mathbf{s}_k^{\text{word}} = \langle h_k^{\text{word}}, \boldsymbol{\mu}_k^{\text{word}}, \boldsymbol{\Sigma}_k^{\text{word}} \rangle$, such that

$$\boldsymbol{\mu}_k^{\text{word}} = \frac{1}{|\Lambda_k|} \sum_{i \in \Lambda_k} \mathbf{v}_i, \qquad \boldsymbol{\Sigma}_k^{\text{word}} = \frac{1}{|\Lambda_k| - 1} \sum_{i \in \Lambda_k} \mathbf{v}_i^k \mathbf{v}_i^{k\,\mathrm{T}} \tag{1}$$

where $\mathbf{v}_i$ represents the 2D coordinate of point i and $\mathbf{v}_i^k = \mathbf{v}_i - \boldsymbol{\mu}_k^{\text{word}}$.

Defining spatiograms over topics is a little more complicated, since topics are not uniquely associated with a distinct set of points, but rather points are shared amongst topics according to the contribution of their assigned words to given topics. This is defined by the topic space matrix $\mathbf{W}$ and thus we can use this to weight points when computing the spatial characteristics of a topic. Given a topic histogram $\mathbf{h}^{\text{topic}}$ its spatiogram $\mathbf{s}^{\text{topic}}$ therefore consists of T triplets $\mathbf{s}_t^{\text{topic}} = \langle h_t^{\text{topic}}, \boldsymbol{\mu}_t^{\text{topic}}, \boldsymbol{\Sigma}_t^{\text{topic}} \rangle$, where the mean and covariance are now given by

$$\boldsymbol{\mu}_t^{\text{tpc}} = \sum_{k=1}^{K} w_{tk} \boldsymbol{\mu}_k^{wrd}, \quad \boldsymbol{\Sigma}_t^{\text{tpc}} = \frac{1}{1 - \beta_t} \sum_{k=1}^{K} \frac{w_{tk}}{|\Lambda_k|} \sum_{i \in \Lambda_k} \mathbf{v}_i^t \mathbf{v}_i^{t\,\mathrm{T}} \tag{2}$$

where $\mathbf{v}_i^t = \mathbf{v}_i - \boldsymbol{\mu}_t^{\text{topic}}$ and $\beta_t = \sum_{k=1}^{K} \frac{w_{tk}^2}{|\Lambda_k|}$. The weights w_{tk} are given by $w_{tk} = W_{tk} h_k^{\text{word}}$ and reflect both the importance of word k through h_k^{word} and its contribution to topic t via W_{tk}; the weights are normalised to sum to 1 over words. Note that it is essential that these weights are not negative, which is why we chose ONMF over the alternatives. Again, since we have two topic spaces, each region will have two spatiograms, one for classification and one for orientation estimation.

To use the spatiograms for classification and regression, we use a similarity measure proposed in [28]. This uses the Battacharyya coefficient to compare spatiogram bins, and a measure of the overlap of Gaussian distributions to compare their spatial distribution. For two spatiograms $\mathbf{s}^A$ and $\mathbf{s}^B$ of dimension M, this similarity is defined as

$$\rho_{AB} = \sum_{m=1}^{M} \sqrt{h_m^A h_m^B} 8\pi |\boldsymbol{\Sigma}_m^A \boldsymbol{\Sigma}_m^B|^{\frac{1}{4}} \mathcal{N}\big(\boldsymbol{\mu}_m^A; \boldsymbol{\mu}_m^B, 2(\boldsymbol{\Sigma}_m^A + \boldsymbol{\Sigma}_m^B)\big) \tag{3}$$

Following [28], we use a diagonal version of the covariance matrices $\boldsymbol{\Sigma}_m$ since it simplifies the calculation.

4.7 Classification and Regression

For classification and regression we use the relevance vector machine (RVM) [34] – a sparse kernel method, conceptually similar to the well-known support vector machine (SVM). Like SVMs, once trained, an RVM uses only a small subset of data, making it very fast even for very large training sets. This is important in our application since for plane detection it must be applied several hundred times per image. Furthermore, unlike the SVM, it gives probabilistic outputs, as opposed to hard decisions, which also proves important for plane detection.

Full details of RVMs can be found in [34]. In summary, given N training samples, $\{y_n, \mathbf{x}_n | n = 1 \dots N\}$, where $\mathbf{x}_n$ is the nth feature vector and y_n its 'target' value, then a predicted target value for a new feature vector $\mathbf{x}$ is based on linear regression amongst the input vectors:

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + b \tag{4}$$

where b is a bias parameter and $k(\cdot,\cdot)$ is a kernel function. The key element is that the individual weights w_n are modelled as normally distributed, whose values are found by maximising the evidence with respect to the distributions' parameters. Many weights tend toward 0, so the above reduces to a sum over a much smaller number of 'relevance' vectors from the training set (in our case over 95% reduction), giving fast prediction. The probabilistic modelling also yields an estimate of the variance (uncertainty) for the prediction.

For classification, the target values are binary (plane or non-plane) and so the prediction is transformed by a logistic sigmoid, mapping to $(0, 1)$ – this is interpreted as the probability of the input belonging to the positive class, which is thresholded to obtain a binary classification. For regression, the target values are 3D orientations, so we need to use the multivariate regression RVM developed by Thayananthan et al. [33]. The input feature vectors are either histograms or spatiograms of words or topics; for spatiograms, we used a polynomial function for the kernel $k(\cdot,\cdot)$ applied to the similarity measure in (3), since in our experiments we found it to perform better than using this measure on its own; thus $k_S(\mathbf{S}^A, \mathbf{S}^B) = \sum_{q=1}^{Q} \rho_{AB}^q$, where we chose the maximum power Q to be 4. The same kernel function was found to perform well for both classification and regression tasks.

5 Plane Detection

In this section, we describe the plane detection algorithm, which identifies planar regions in images and estimates their orientation. As illustrated in Fig. 4, we do this by applying the plane recognition algorithm at different locations over the image, and use this to estimate planarity at individual points, and after clustering and smoothing we are able to extract individual planar structures. Example results from each stage are shown in Fig. 5.

The recognition algorithm, as explained above, uses a discrete set of points, and so the density of such points determines the segmentation resolution that we can expect. In effect, we aim to group these points into planar and non-planar regions based on their spatial adjacency and compatibility in terms of planar characteristics; since these points are the centroids of the image patches used to describe the image, our point-based segmentation is effectively segmenting the image itself. We present two versions of the algorithm: first the method using DoG salient points, then in Section 5.3 we show how a denser, regular grid at multiple image scales can be used to increase coverage of the image. Hereafter, 'point' is used to refer to such a salient point or grid vertex as appropriate.

5.1 Location Sampling

The first stage of the algorithm is to apply plane recognition (PR) (Section 4) at multiple overlapping locations in the image, to sample possible locations where planes might be. We use a set of up to 100 regions per image, centred over a subset of points. These regions are circular with a fixed radius (50 pixels in the experiments) and all points within such a region are used as input to one invocation of PR, giving planar/non-planar classifications (and orientation estimates) at these locations.

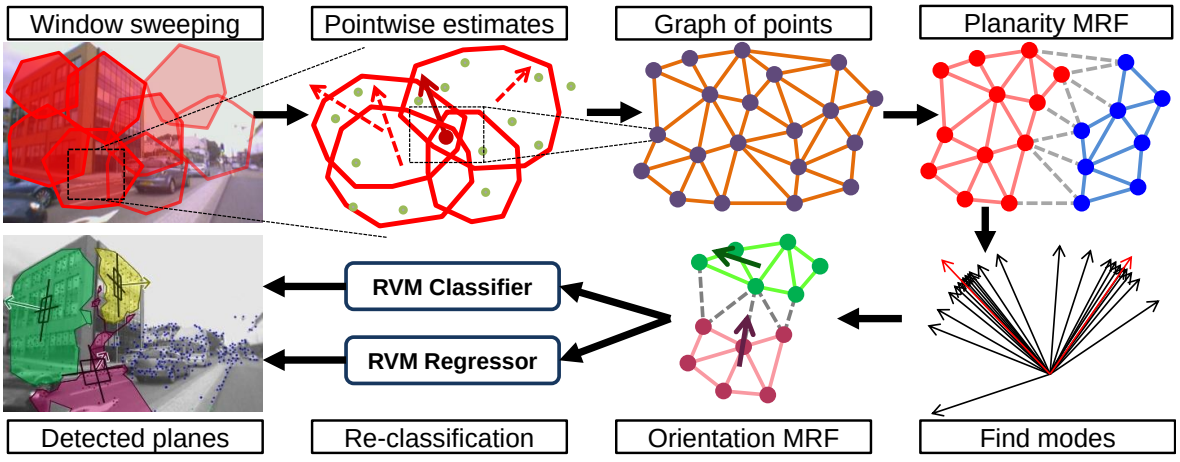


Fig. 4. Main steps of the plane detection method (using DoG salient points).

To gather a more appropriate training set for this stage (the regions are a different size and shape compared to Section 4), we use our whole-image ground-truth data set (examples of which can be seen in Fig. 10), and extract circular regions of a given radius, in the same way as for location sampling. Labels for the regions are assigned according to the majority class and mean orientation of the points within the region, according to the ground-truth..

As illustrated in Fig. 4, after applying PR to these sampled regions we obtain multiple recognition results for a given point, from all the overlapping regions in which it lies, i.e. all those regions for which it was used for recognition. After discarding regions for which classification certainty is too low, we assign a single class and (if planar) an orientation estimate to the point: each point i is given an estimate of its probability of being on a plane, denoted r_i, and of its normal vector $\mathbf{d}_i$, which are calculated using the median and geometric median to give a degree of robustness to outliers. The result is an estimate of planarity for each point, such as in Fig. 5b, which we refer to as the local plane estimate (LPE). These pointwise estimates are informative, but clearly do not constitute a plane detection, nor are they necessarily accurate – but they are sufficient for segmenting planes from each other, as we describe next.

5.2 Segmentation

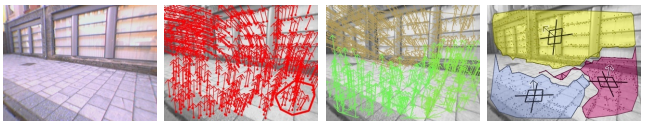
The goal of the segmentation stage is to take the points in the local plane estimate (above), and separate them into distinct planar or non-planar regions. This is achieved in three steps: first, to cluster the labels assigned to points to obtain a discrete set of assignable labels; assign each point its most likely label; and then to extract connected regions of points with the same label. While a number of segmentation algorithms could be employed to achieve this, the problem is naturally expressed as a simple MRF on a graph connecting the points.

The segmentation of planes from non-planes, and into planes of different orientations, is done separately, since different criteria are used for the two stages (classification probabilities and orientation estimates, respectively), and they act on different subgraphs of the point set (orientations are usually not defined for regions deemed non-planar). A joint segmentation should be possible, but we present details of the simpler model here.

For separating planes from non-planes, no clustering is necessary on the labels, since there are only two possibilities, being 1 and 0, plane and non-plane respectively. Each point i is set to the value $p_i \in \{0, 1\}$ which is closest to its observed value r_i (the median of the sampled estimates, as above). To extract contiguous regions from these points, a graph is created, using either a Delaunay triangulation or a regular grid, depending on the saliency type used. To create a smoother segmentation, and deal with any noise or anomalous assignments, we apply smoothing based on iterative conditional modes (ICM) [2], so that labels depend upon points' observations plus their neighbours' labels. This is formulated as a simple MRF: let p represent a configuration of the field, where each node $p_i \in \{0, 1\}$ represents the label of point i. We then seek the optimal configuration p^*, defined as $p^* = \arg\min_p U(p)$, where $U(p)$ is the posterior energy of the MRF:

$$U(p) = \sum_{i \in S} V_1(p_i, r_i) + \alpha_P \sum_{i \in S} \sum_{j \in \mathcal{N}_i} V_2(p_i, p_j) \quad (5)$$

where S is the set of all nodes and $\mathcal{N}_i$ are the neighbours of node i. The functions V_1 and V_2 are the single site and pair site clique potentials, respectively, where $V_1(p_i, r_i) = (p_i - r_i)^2$ and V_2 has value 0 if p_i and p_j are equal, 1 otherwise, and α_P is a weighting parameter to balance the effects of the unary and pairwise potentials. This MRF is easily optimised using ICM; while more efficient alternatives such as graph cuts exist [21], we find this simpler approach sufficient,



(a) (b) (c) (d)

Fig. 5. Outputs from plane detection: from the input image (a), we apply plane recognition over the image to obtain a point-wise estimate of orientation (b). This is segmented into distinct regions (c), from which the final plane detections are derived (d).

and it generally converges within a few iterations. Thus each node is set to its most likely value (plane or not), given the estimates from each of the sampled regions and a smoothness constraint imposed by its neighbours. Planar regions are extracted from this by connected component analysis.

Subdividing planar regions based on their orientation estimates is a little more complicated, as the labels belong to the continuous range of normal vectors, rather than simply two classes. We deal with this by making the assumption that the image consists of a finite number of planar surfaces, consisting of sets of nearby points which have the same normal (i.e. piecewise constant regions). We find the dominant orientations using mean shift clustering [6], which gives us the modes of the kernel density estimate of the distribution of orientation estimates in the image. Normal vectors are represented as two angles (θ, ϕ), and a Gaussian kernel with a bandwidth of 0.2 radians is used (a value chosen by cross-validation experiments omitted here due to space constraints).

After clustering, we can assign discrete labels to the points, according to their observations. As above, we also want to take into account the labels of neighbouring points, to impose smoothness on the resulting regions, so we again formulate the label assignment as a simple MRF, defined on the subgraph comprising only the points which were deemed to be in planar regions, denoted by the set $S' \subset S$. Similar to the above, let $\mathbf{n}$ represent the configuration of this MRF, with $\mathbf{n}_i \in \mathbb{R}^3$ representing the 3D normal at node i. We wish to obtain $\mathbf{n}^* = \arg\min_n E(n)$, where the posterior energy $E(\mathbf{n})$ is

$$E(\mathbf{n}) = \sum_{i \in S'} F_1(\mathbf{n}_i, \mathbf{d}_i) + \alpha_O \sum_{i \in S'} \sum_{j \in \mathcal{N}_i} F_2(\mathbf{n}_i, \mathbf{n}_j) \quad (6)$$

where both clique potential functions F_1 and F_2 return the angle between two vectors in $\mathbb{R}^3$, and where $\mathbf{d}_i$ is the observed normal for point i (the geometric median of its regions' estimated normals). α_O is again a weighting parameter. We generally set both weighting parameters to be the same, $\alpha_P = \alpha_O = 1$; the effect of this is discussed in Section 6.4.

After optimising this second MRF, we extract the set of disjoint planar regions, formed from connected components of the graph which were assigned the same orientation label (if two disconnected regions have the same orientation, they are considered different, but parallel, planes). These plane segments are passed through the plane recognition algorithm once more, to verify they are planar and update their orientation, so that the final values are based on the regions themselves, rather than the surroundings of all points in the region. Generally, we find that the orientations do not change substantially, suggesting that pointwise plane estimates are a good approximation with which to segment planes. These regions are no longer the same shape as the locally sampled regions – we therefore train a second pair of RVMs from data obtained by running the full segmentation algorithm on the original training set of ground truth images, which will be more similar to the final regions. This improves orientation accuracy of the final planar regions by several degrees. Non-planar segments are discarded, as are any segments which are subsequently re-classified as non-planes (leaving a set of non-planar points in the image).

5.3 Grid-based Detection

The method as described above (and in [14]) uses the DoG saliency detector to select points in the image around which to create descriptors, which is a convenient way of avoiding regions with no texture. Unfortunately, this prevents our method from working in a number of situations, such as road surfaces with very little texture information. However, there is no reason why a different, denser set of points cannot be used. We investigated this by replacing the DoG detector by simply using all pixels in a regular grid. Using every single pixel would increase the computational burden significantly, so we compromise with a grid with a spacing of ten pixels.

To retain the benefit of using image patches at multiple scales, we use a multi-resolution grid, where grids at a coarser scale use larger image patches. We use four grids of different scales, each at twice the separation of the level below (spacings of 5, 10, 20 and 40 pixels, for patch sizes of 10, 20, 40 and 80 pixels respectively). The subsequent parts of the algorithm operate in the same way, using descriptors for all grid points which fall into a certain radius of selected points to create sample regions, as before. We also omit gradient descriptors in regions where there is no texture, relying solely on colour. The main difference is that the Delaunay triangulation is no longer required to create the graph – each point is connected to its 4-neighbours. Furthermore, while patches at larger scales are used for providing information to the classifiers, it is not necessary to include these duplicate points in the graph.

The effect of this modification on the algorithm is that it is now able to extend to more of the image (though we concede that in regions devoid of texture,

accuracy is likely to fall), and thus addresses one of the principal ways in which existing methods (most notably [19]) achieve superior coverage of the image.

6 Results

This section presents results of experiments to evaluate the proposed algorithms. First, we look at performance of the plane recognition algorithm on individual image regions; before showing the results of experiments to evaluate the full plane detection method, both against our own ground truth data, and by comparing with prior work.

6.1 Plane Recognition

The data we used for evaluating the plane recognition algorithm consist of regions extracted manually from images (we are not using the whole image), labelled with the true class (plane or non-plane) and orientation (normal vector), as described above. We used two datasets, for training and testing. The training set was used for cross validation, to evaluate the accuracy and consistency of the method, and to investigate performance using different representations and parameter values, before training the full algorithm. This consisted of 556 regions captured by a 320×240 pixel calibrated webcam: we use a relatively low resolution camera to match our intended applications in real-time SLAM [16], where image size has to be kept small due to computational constraints. These images were reflected, to give 1112 regions, and warped, giving a total of 10155 regions (we do not test on the warped regions, and ensure that warped versions of a test region are never in its training set). The second dataset, of 690 images, was gathered in a distinct location to the first, to ensure there was no overlap, with which we evaluate the generalisation ability of our algorithm (having trained on the first dataset). These images are of the same format and with the same intrinsic camera parameters.

6.1.1 *Cross-validation*

In the cross-validation experiments we investigated the effects of using different representations, features, and vocabulary size. Statistics were obtained over ten independent runs, each using 5-fold cross validation (these comparisons use the mean orientation error and its standard deviation, rather than the median). First, we compared orientation error when using words, topics, histograms and spatiograms as the vocabulary size is varied. We used only the gradient vocabulary and the reflected dataset (without warps) to make the effects of the other parameters clear. As Fig. 6 shows, spatiograms outperform histograms, and topics are beneficial, especially for larger vocabulary sizes. Given that small vocabularies are likely to constrain the method, these results confirm the advantages of using topic spatiograms.

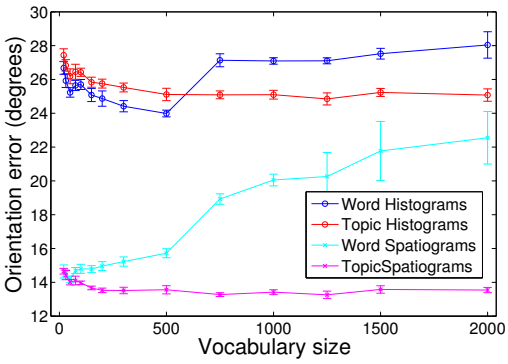


Fig. 6. Comparison of mean orientation estimation error using word and topic histograms and spatiograms while varying vocabulary size.

	Gradient	Colour	Grad. & Col.
Classification Acc. (%)	86.5 (1.8)	92.5 (0.5)	93.9 (2.8)
Orientation Err. (deg)	13.1 (0.2)	28.4 (0.3)	17.9 (0.7)

TABLE 1
Comparison of average classification accuracy and mean orientation error when using gradient and colour features. Standard deviations are shown in brackets.

We next compared performance when using gradient features and colour features. Table 1 shows that classification using colour alone gives better performance than gradient features, but combining the two is better overall. However, as expected, colour is much worse for estimating orientation, and combining the two offers no improvement. Since we use separate RVMs for the two steps, using different features is not a problem, and so we maintain separate spatiogram descriptors for classification and regression.

It may be thought that part of the success of spatiograms is because they, unlike histograms, are

	Cut Hist.	Cut Spat.	Hist.	Spat.
Class. Acc. (%)	75.3 (1.0)	84.4 (0.9)	77.3 (1.0)	87.9 (1.0)
Orient. Err.	26.6 (0.2)	17.0 (0.3)	24.9 (0.2)	13.3 (0.1)

TABLE 2
Comparison of performance for histograms and spatiograms on regions cut to be uniformly circular, compared to the original shaped regions.

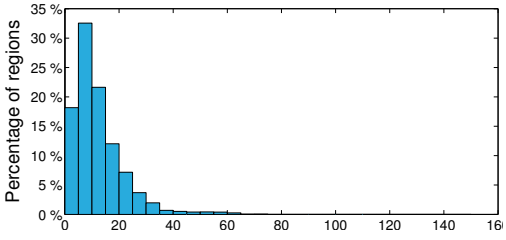


Fig. 7. Histogram of orientation errors for the recognition algorithm, showing that the majority of regions are given an orientation estimate with low error.

able to implicitly encode region shape, and that these shapes, being manually defined by a human, may offer cues to the orientation of the plane. This is a problem, since for regions obtained in other ways, such cues would not be available, and may bias the classifier by falsely predicting planes due to coincidental region shape. To investigate, an experiment was carried out where all regions were cut to being circular in shape. Spatiograms still significantly outperform histograms, for both classification and regression, as shown in Table 2 – while the cut regions exhibit worse performance, there is still a clear increase in performance when using spatiograms, indicating the shape of regions is not a crucial factor behind the benefit of spatiograms.

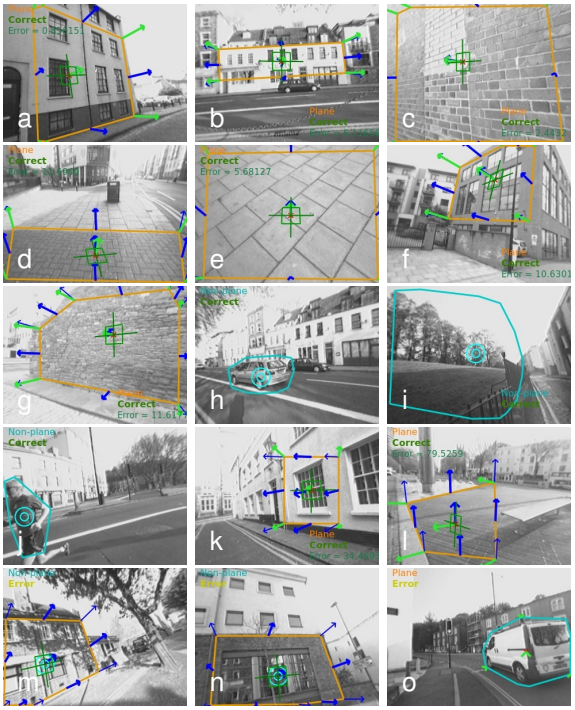


Fig. 8. Example outputs of plane recognition, showing correct classification (a-j) and good orientation estimation (a-g), plus some failure cases: poor orientation estimate (k,l), misclassification as non-plane (m,n), and misclassification as planar (o). Orange/cyan boundaries denote ground-truth plane/non-plane respectively; those classified as planes have green arrows (estimated orientation), ground-truth orientation is drawn with blue arrows.

For the final test with cross-validation, we used gradient and colour and features for classification, in vocabularies of 400/300 words respectively, and gradient features only for regression, compactly represented with spatiograms over topics. We also added the reflected and warped images, to supplement the training set, which was previously found to improve accuracy [13]. Cross validation was run ten times, and we report the mean results. We observed a mean classification accuracy of 95% (standard deviation 0.49%) and a median orientation error of 9.8°. We report the median accuracy for orientation estimation, rather than the mean, because this is more robust to outliers and gives a better indication of performance when the distribution is skewed. This is clearly the case here, as shown in Fig. 7, which plots the distribution of orientation errors. Some outlying regions have large errors, but a significant number are under 15° (72%) and under 20° (84%).

6.1.2 *Independent Data*

To evaluate the ability of the algorithm to generalise, we use the second, independent dataset, gathered from a different (but similar) urban location. While there may be some coincidental similarity in appearance, there is no overlap between the training and test sets, unlike in the cross-validation tests. This dataset consists of 690 image regions, divided equally between planes and non-planes. In this experiment we trained the algorithm using the full test dataset from above, including the reflected and warped images, and used the settings described in the previous section. The results we observed were an average classification accuracy of 91.6%, and a median orientation error of 10.6°, suggesting that the algorithm is capable of generalising well to new environments.

Figure 8(a-j) shows examples of successful plane recognition, from the independent data. Correctly classified planes and their orientation are indicated by green arrows (ground truth is shown in blue) while correctly identified non-planar regions are indicated by cyan circles, including vehicles, foliage and people. Note in particular the variation in appearance of the planar regions, including both regular and irregular texture, demonstrating the generality of the algorithm.

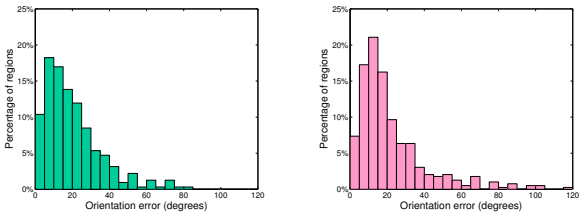
We also show some examples of where the algorithm fails Fig. 8(k-o). These include a mix of poor orientation estimates and false positive and negative classifications, due either to insufficient texture or ambiguity in line orientation, for example. However, these are in a minority; most regions are accurately classified, with low orientation error.

6.1.3 *Comparison to KNN*

We note that it is possible to replace the RVM with K-nearest neighbour (KNN) classifiers, finding neighbours using the spatiogram similarity measure. In cross-validation experiments, we obtained a mean classification accuracy of 92% and a median plane orientation error of 10.7°, close to that reported above; although because it is necessary to compare with all the training examples, the KNN is too slow to use in practice. Nevertheless, this is interesting since it confirms the success of the method is not due to some property of the RVM, but because the features and similarity measures we use are able to match appropriate structures. Further details and examples can be found in [13], [15].

6.2 Plane Detection Results

This section describes the experiments we performed on the plane detection algorithm. For these experiments, we use two datasets: the training set, comprising 439 images taken from urban locations, including many examples of both planes and non-planes; and an independent test set, of 138 images from a separate urban location, such that there is no physical overlap between the two. This is an expanded version of the data previously used in [14], so results may differ. We have made these data available online[1] to facilitate future comparisons.



(a) Saliency Detection (b) Grid

Fig. 9. Distribution of orientation errors for plane detection on previously unseen data.

When testing on the independent test set, we obtained a mean point classification accuracy of 83.6%, and a median orientation error of 16.2°, measured over the detected regions (the distribution remains heavily skewed toward lower errors, as seen in Fig. 9(a)). Note that unlike in the plane recognition case (Section 6.1), the test and training regions are no longer guaranteed to be wholly planar. As such we believe these detection results are very reasonable given the difficulty of the task and that no explicit use is made of geometric information.

We also measure how much of the image is covered by the detection. This value is calculated as the total area of the image, in pixels, which falls under any of the patches used to create descriptors. This is useful for evaluating how capable the detection algorithm is of finding planes from across the whole image; and also emphasises the point that we are not simply labelling individual points, but grouping together regions of the image. In this experiment, we measured a coverage of 75%.

Figure 10 shows a selection of results, alongside the intermediate local plane estimates (Section 5.1) and ground truth. Example 10(c) shows that it is quite capable of detecting planes in environments where there are dominant vanishing lines; but example 10(d) is important since it shows it can also cope in the absence of any obvious geometric structure, where such methods would fail. Note also Fig. 10(b) (and Fig. 1), where non-planar areas are successfully segmented from the planar surfaces.

1. Our dataset can be found at www.cs.bris.ac.uk/~haines

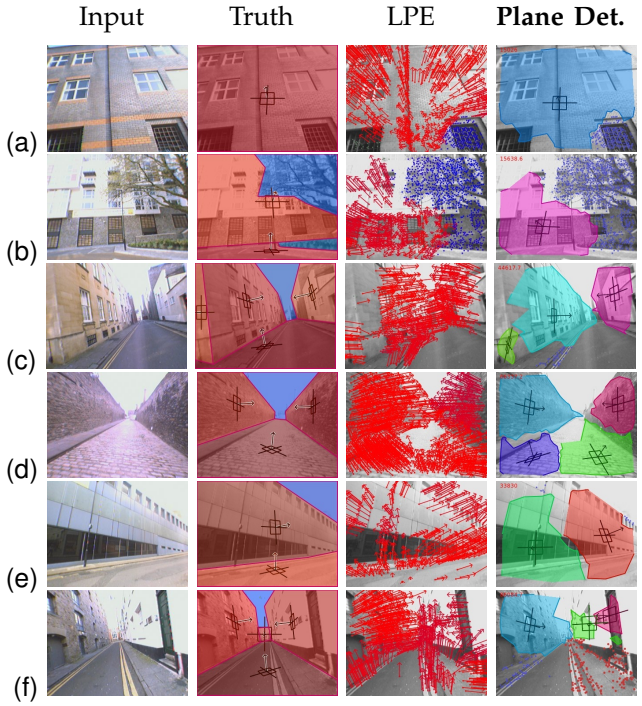


Fig. 10. Examples of plane detection. In the ground truth, red regions are planes, blue are not. In the detection results, groups of planar points are enclosed in coloured regions, displaying their orientation; individual coloured points are non-planar.

We also show some examples where our algorithm fails. Figure 10(e) shows an example where one plane has been split in two, and extends over the ground plane, while Fig. 10(f) shows the ground plane being completely missed, and planes on the right being given the wrong orientation.

6.3 Multi-resolution Grid

In Section 5.3 we described an alternative method of performing plane detection, by using a dense multi-resolution grid of points instead of DoG saliency detection. This allows the method to cover much more of the image, including potentially featureless areas.

To compare this with the original version, we evaluated this new algorithm on the independent dataset. It gave a classification accuracy of 81.6%, and a median orientation error of 16.3° (histogram of errors in Fig. 9(b)). These results are a little worse than the DoG method, but we believe this constitutes good performance – especially since this means the detection is now covering almost the entire image (94% by area).

Since we can now cover almost the entire image, we can calculate performance measures for plane detection itself, in terms of precision and recall. Precision is defined as the proportion of detected planes which are true detections, i.e. those for which at least 70% of their points are in ground truth planes. Recall is similarly defined as the proportion of ground truth

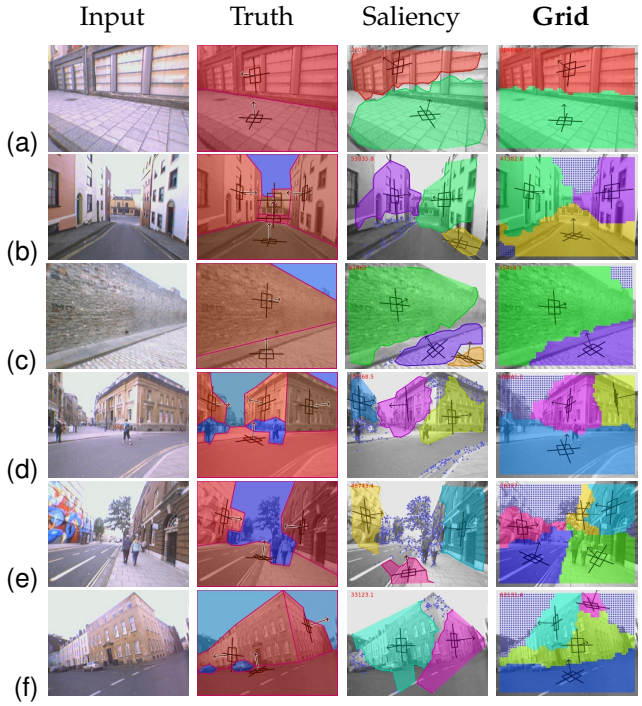


Fig. 11. Examples of using the grid-based whole-image method, compared to ground truth and the original saliency-based version.

planes which are successfully detected, where a successful detection is one for which at least 70% of its points lie within a detected plane. For this experiment, we obtained a precision of 0.89 and a recall of 0.67.

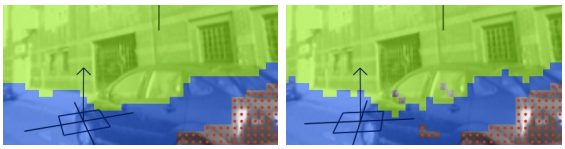
To illustrate the effect of this alternative method, we show example results in Fig. 11. Some regions which the original algorithm misses (especially surfaces of roads) are indeed detected as being planar, and can be given correct orientations, as shown in Fig. 11(d) for example. The resulting detections give a more complete description of the image, giving a better sense of the structure of the scene, as in Fig. 11(b).

In other instances, this new algorithm fails, and spreads planes across inappropriate regions of the image, or assigns a visibly erroneous orientation. Figure 11(e) shows many wrong planes across the image, while the more conservative DoG version gives the better result. In Fig. 11(f) the grid-based version has successfully found the road surface, but the two faces of the building have been merged.

Despite this, it is clear from the examples given that the grid-based detector can offer some advantages over the original, and may be beneficial in some situations where area of coverage takes precedence to metric accuracy. This difference is also useful when comparing to existing methods.

6.4 Smoothing

In Section 5.2 we discussed how a MRF can be employed to smooth the assignment of planarity and orientation labels to the points, before extracting regions.



(a) With smoothing (b) No smoothing

Fig. 12. The effect of smoothing with iterative conditional modes – taking point neighbourhoods into account creates regions with smoother boundaries, and avoids isolating small groups of points.

The parameters α_P, α_O control the relative influence of the unary and pairwise terms on the smoothing. In our experiments, we found that different non-zero values have very little influence on the result. If we set $\alpha_P = \alpha_O = 0$, this removes the influence of neighbouring points on the segmentation, essentially removing the MRFs. This gives similar results, but the regions are less smooth: as Fig. 12 shows, boundaries are more jagged without smoothing, and some pixels are left out of any regions. It is an interesting aspect of our method, that even if the MRF is removed completely, plane and non-plane regions can still be detected simply by analysing connected components in the graph, due to the processing performed before giving the data to the MRF. However, while using the MRF does not significantly affect the accuracy measures quoted above, it does lead to an improved presentation of planar regions, which could be advantageous when using these regions in other tasks [16].

6.5 Comparison to Prior Work

As outlined earlier, a common way to achieve plane detection is to use cues such as vanishing lines and rectilinear structure; however, we do not compare to these since they operate in very different ways and only under limited conditions. While they may give superior estimates when sufficient orthogonal structure is present, but are likely to fail in more complex scenes (for example in Fig. 1).

The closest prior work is that of Hoiem, Efros and Hebert [19] (henceforth we shall refer to this as HEH), where planar surfaces are detected using a variety of features, in a machine learning framework. Unfortunately, comparison to this is not straightforward, since the two algorithms are very different in their intentions. HEH does do a form of plane detection (which is simple enough to compare); but it does not estimate the orientation of planes, instead categorising them into a discrete set of orientation classes.

We continue to use the percentage of points assigned to the correct class as the measure of classification accuracy, but to measure orientation estimation, we alter our method to quantise the orientation estimates into the same classes. We compare HEH to

ours by measuring accuracy only at the points we use (the DoG points, or the regular grid), since we cannot compare our algorithm at every pixel. This quantisation undoes one of the main benefits of using our method, so such a comparison of accuracy is not necessarily meaningful. Nevertheless, we present such a comparison as a useful visualisation to the reader of the ways in which the two algorithms can be made to perform a similar task.

To run HEH on our data, we used Matlab code provided by the authors[2]. We used this to re-train their classifiers on our training set, after quantising orientations to discrete classes (note that we do not have annotations for the 'porous' or 'solid' classes, and set everything non-planar to 'sky'). We found this gave better performance on our data than the pre-trained classifiers they provide.

We compared HEH to both versions of our method, using salient points or the dense grid. We found that the two did indeed perform similarly, with HEH achieving accuracies of 89% and 84% for plane and orientation classification respectively, compared to our grid-based method at 83% and 80%. Using the DoG method, we achieved a similar accuracy, of 84% and 80%, which is almost exactly the same as theirs when sampled at the same points, suggesting that focusing on more highly textured locations benefits our method. Their method appears to out-perform ours by up to 5% in accuracy, though we emphasise that the algorithms are set up to do very different things, and this test shows our algorithm is able to perform a similar task to their method, even though it was designed for something different. We also highlight that our method achieves relatively better performance when focusing on textured regions, and that we can opt to detect planes over the entire image, but at the cost of accuracy.

Figure 13(a) illustrates how we re-interpret the output of HEH to show plane detection results, for a clearer comparison: we treat classification into sky, porous or solid classes as being non-planar, drawn in blue; and the others (left, right and forward subclasses of vertical, and the support plane) are planar, which are drawn in colours according to their orientation, with arrows overlaid to show orientation. We display our results and ground truth in the same manner, as Fig. 13(b) shows, where colours and symbols surround each (grid) point, and our orientation vectors have been quantised into the nearest orientation class.

Example results are shown in Fig. 14, comparing our grid-based method (we found this to be much better able to approach the kind of whole-image segmentation offered by HEH, due to its greater image coverage) to HEH and the ground truth. Our method appears capable of performing a very similar kind of orientation classification to HEH, although it performs

2. www.cs.uiuc.edu/~dhoiem/

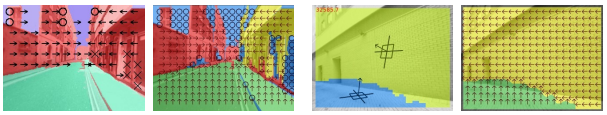


Fig. 13. Illustrations of how the output of HEH (a) and our (grid based) method (b) are re-drawn to show quantised plane detection.

it in a different way. In some cases, our method is better able to deal with large planes (e.g. Fig. 14(b)); and by being able to estimate orientation, somtimes distinguishes planes of different orientation, whereas HEH may merge them together, like in Fig. 14(a). On the other hand, HEH is better able to deal with small or distant regions (Fig. 14(e)).

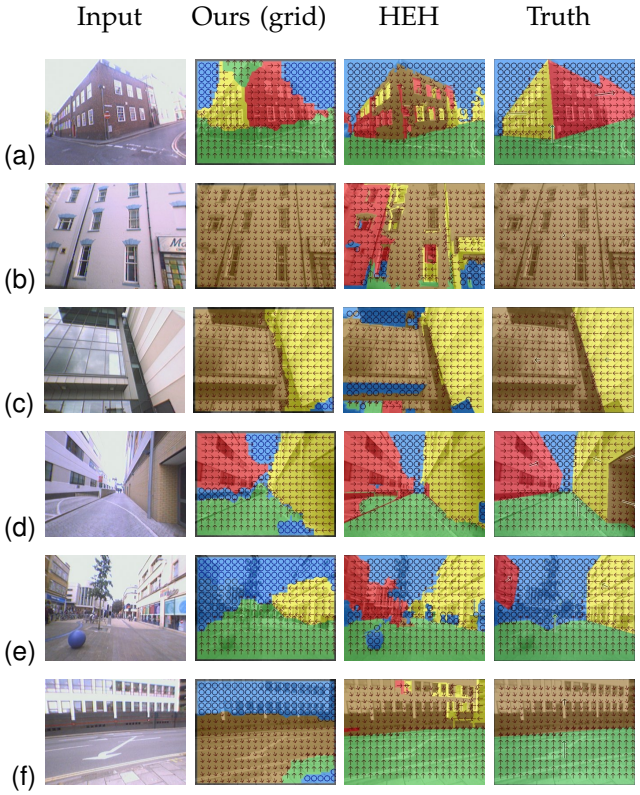


Fig. 14. Some example results, comparing our grid-based algorithm to HEH and ground truth.

7 Conclusions

We have shown that it is possible to learn the relationship between appearance and structure in single images, and presented a new algorithm to detect planes, which can for the first time estimate a 3D orientation. The approach comprises a method to estimate the planarity and orientation of individual regions, based on learning from a training set. This is then used in a plane detection algorithm, that does

not require a priori region segmentation or knowledge of plane boundaries. Our algorithm can detect planes with good accuracy compared to labelled ground truth, and gives comparable segmentations to the most similar work [19].

The plane detection works by repeated sampling of windows to recover individual planes; however, this makes it unable to deal with small planar regions. An avenue of future work, therefore, would be to incorporate edge or contour information, which can be beneficial in scene layout estimation [20]. A similar technique could also be applied to relative depth estimation [32] – to improve the fidelity of plane detection, or to use alongside plane detection for more sophisticated interpretation of images.

ACKNOWLEDGMENTS

We would like to thank the UK Engineering and Physical Sciences Research Council for funding this work.

REFERENCES

[1] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin. Fast automatic single-view 3-d reconstruction of urban scenes. In *Proc. European Conf. Computer Vision*, 2008.

[2] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302, 1986.

[3] S.T. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, 2005.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022, 2003.

[5] S. Choi. Algorithms for orthogonal nonnegative matrix factorization. In *Proc. IEEE Int. Joint Conf. Neural Networks*, 2008.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[7] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Int. Journal of Computer Vision*, 40(2):123–148, 2000.

[8] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[9] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, 2005.

[10] J. Gårding. Direct estimation of shape from texture. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1202–1208, 1993.

[11] R.L. Gregory. Knowledge in perception and illusion. *Philosophical Trans. the Royal Society of London. Series B: Biological Sciences*, 352(1358):1121–1127, 1997.

[12] F. Guo and R. Chellappa. Video metrology using a single camera. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(7):1329–1335, 2010.

[13] O Haines. *Plane Detection from Single Images*. PhD thesis, University of Bristol, 2013.

[14] O. Haines and A. Calway. Detecting planes and estimating their orientation from a single image. In *Proc. British Machine Vision Conf.*, 2012.

[15] O. Haines and A. Calway. Estimating planar structure in single images by learning from examples. In *Proc. Int. Conf. Pattern Recognition Applications and Methods*, 2012.

[16] O. Haines, J Martínez-Carranza, and A Calway. Visual mapping using learned structural priors. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2013.

[17] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[18] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, 2006.

[19] D. Hoiem, A.A. Efros, and M. Hebert. Recovering surface layout from an image. *Int. Journal of Computer Vision*, 75(1):151–172, 2007.

[20] D. Hoiem, A.N. Stein, A.A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *Proc. Int. Conf. Computer Vision*, 2007.

[21] P. Kohli and P. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(12):2079–2088, 2007.

[22] J. Košecká and W. Zhang. Extraction, matching, and pose recovery based on dominant rectangular structures. *Computer Vision and Image Understanding*, 100(3):274–293, 2005.

[23] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, volume 2, pages 2169–2178. IEEE, 2006.

[24] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[25] C. D. Manning and H. Raghavan, P. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[26] J. Martínez-Carranza and A. Calway. Efficient visual odometry using a structure-driven temporal map. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2012.

[27] B. Mičušík, H. Wildenauer, and J. Košecká. Detection and matching of rectilinear structures. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, 2008.

[28] C. Ó Conaire, N.E. O'Connor, and A.F. Smeaton. An improved spatiogram similarity measure for robust object localisation. In *Proc. IEEE Int. Conf. Acoustics, Speech & Signal Processing*, 2007.

[29] G. Orbán, J. Fiser, R. Aslin, and M. Lengyel. Bayesian model learning in human visual perception. In *Proc. Advances in Neural Information Processing Systems*, 2006.

[30] E. Prados and O. Faugeras. Shape from shading. In *Handbook of mathematical models in computer vision*, pages 375–388. 2006.

[31] Srikumar Ramalingam and Matthew Brand. Lifting 3d manhattan lines from a single image. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2013.

[32] A. Saxena, M. Sun, and A.Y. Ng. Make3D: learning 3D scene structure from a single still image. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2008.

[33] A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla. Multivariate relevance vector machines for tracking. In *Proc. European Conf. Computer Vision*, 2006.

[34] M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

[35] A. Torralba and A. Oliva. Depth estimation from image structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(9):1226–1238, 2002.

Osian Haines graduated from Cardiff University in 2008 with a BSc in Computer Science with Vision and Graphics. In 2013 he received a PhD from Bristol University, where he continues to work as a post-doctoral research assistant. His main research interests are in single image understanding and computer vision for robotics.

Andrew Calway received his PhD from Warwick University in 1989. He was a Royal Society visiting researcher in the Computer Vision Laboratory at Linköping University, then lectured at Warwick and Cardiff University. In 1998, he joined Bristol University, where he is now a reader in computer science, with research interests including image processing and computer vision.