Fusing Intertial Data with Vision for Enhanced Image Understanding

Osian Haines, David R. Bull, and J.F. Burn

University of Bristol
research@osianh.com
{dave.bull,j.f.burn}@bristol.ac.uk

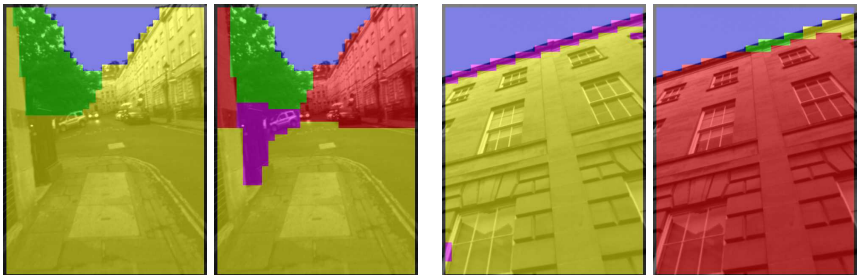
Abstract. In this paper we show that combining knowledge of the orientation of a camera with visual information can be used to improve the performance of semantic image segmentation. This is based on the assumption that the direction in which a camera is facing acts as a prior on the content of the images it creates. We gathered egocentric video with a camera attached to a head-mounted display, and recorded its orientation using an inertial sensor. By combining orientation information with typical image descriptors, we show that segmentation of individual images improves in accuracy compared with vision alone, from 61% to 71% over six classes. We also show that this method can be applied to both point and line based features from the image, and that these can be combined together for further benefits. Our resulting system would have applications in autonomous robot locomotion and guiding visually impaired humans.

Keywords: Vision guided locomotion, segmentation, image interpretation, scene understanding, inertial sensors, Oculus Rift, mobile robotics.

1 Introduction

The ability to safely traverse rough terrain is crucial to the survival of almost all land animals, and is a crucial requirement in order to hunt prey, forage for food, escape predators, find mates, migrate, and so on. Vision is a very important sense for this, and can provide a rich depiction of the surrounding world; but vision is rarely used in isolation, and sound, scent and touch all provide important information too. Another very important source of information is the vestibular system, allowing accelerations and absolute orientations to be perceived independently of visual or other cues [1], and is crucial for balance and normal locomotion [29]. Vestibular information becomes even more important when vision is impaired [7], and its absence can lead to problems in interpreting visual information [30]. There is also some evidence that the central nervous system dynamically controls the relative importance of visual and vestibular signals [7], and that reciprocal inhibition of visual and vestibular signals allows perception of self-motion in situations with conflicting stimuli [4], showing that there is significant and important interaction between the two senses.

There are a large number of applications in computer vision and robotics where visual information is used to guide wheeled or legged vehicles over unknown terrain (see for example [8, 20]). However, the use of orientation information alongside visual information has been less well studied, despite the apparent biological motivation for doing so. Fusing information from these sensing modalities offers great potential for assisting in tasks relevant to locomotion, for example the structural interpretation of image content which we consider in this work. This is based on the observation that the content of an image typically changes in relation to its real-world orientation – for example a downward pointing camera tends to be looking at the ground, while a sideways facing camera can expect to see a combination of walkable terrain and obstacles. This information alone is not sufficient for predicting image content of course, since it disregards any specific information about the current scenario; but it can serve as a useful prior for the kinds of structures to be expected, when used in combination with visual information.



(a) (b)

Fig. 1. Typical results of our algorithm, showing how segmentation results using only vision (left) can be improved by taking into account the camera orientation (right). In both examples knowledge of the camera orientation avoids misclassifying vertical walls as ground (yellow). See Fig. 7 for full color legend. All images are best viewed in color.

Our work develops these ideas and presents the first method, to our knowledge, for combining visual and vestibular information for semantic image segmentation. Using this we show that by combining camera orientation, measured with an inertial sensor, with visual features extracted from images from the camera, we can achieve improved performance in an image segmentation task. The ultimate aim of this work is to build a system enabling autonomous locomotion by legged robots. In order to work towards this goal we focus on developing a method for guiding humans through urban landscapes. Not only is this a convenient test-bed for evaluating vision guidance algorithms without the constraints of robot locomotor capability, but it also demonstrates a potential application in guiding visually impaired humans, where knowledge of the scene structure is of great importance [28]. With this in mind we developed an algorithm which segments images into relevant structural regions, such as the walkable ground

region, impassable obstacles and intermediate surfaces such as stairs, and displays the result through a head-mounted display unit. Examples outputs of the algorithm are shown in Fig. 1.

The next section discusses related work in the field. A brief overview of our method is given in Section 3, followed by Section 4 which describes the data acquisition process. Section 5 gives full details of how our algorithm works. We then present extensive results and examples in Section 6, before concluding in Section 7. Please note that this paper is an extended version of our earlier work [13], incorporating new results and examples.

2 Related Work

Using inertial sensors has been known to improve performance in a variety of computer vision tasks. One of these is visual simultaneous localisation and mapping, in which the pose of a camera with respect to a map, and the unknown map itself, must be recovered. The pose estimate given by an inertial sensor can be fused with that derived from vision to improve robustness [22] and help to mitigate scale drift [24]. A rather different example from [15] uses inertial information for blur reduction, by using estimates of the camera's motion derived from an inertial sensor during an exposure to guide deconvolution.

The work of Hoiem at el. [14] is more closely related to ours, in that images are segmented into geometrically consistent regions. As well as visual features, this uses the position of a segment within the image as a feature during classification, to learn that sky occurs toward the top of the image, for example – although in this work the camera is assumed to be in an upright position with no roll. Visual segmentation can be enhanced using other 3D information – for example using features extracted from a point cloud to help classify objects in road scenes [26]; or by jointly segmenting individual video frames and labelling structures in a 3D reconstruction from those frames [18]. While the orientation of the camera may be implicitly included in these methods via the 3D map, this is not directly investigated, and furthermore is estimated from the image stream itself.

The use of inertial data for terrain classification was investigated by [25], where the inertial data themselves are used as features to encode the vehicle vibration and accelerations for different terrains, in order to predict the terrain type over which the vehicle traverses. This bears some similarity to our approach, in that inertial data is being used for classification, but it does not attempt to make use of the relationship between class and pose.

While these show interesting uses of information not directly present in the image to aid labelling, they are not making use of the information potentially provided by the camera orientation itself. Similarly, while some of the above mentioned works use inertial data to aid vision tasks, this is generally in a purely geometric sense, and they have not exploited the relevance to semantic attributes in the image. We investigate ways to do this in the following sections.

3 Overview

In this paper we present an algorithm which takes as input a single image and its associated 3D orientation, measured with an inertial measurement unit (IMU), and produces a segmentation of the image into distinct regions, corresponding to classes relevant to the task of locomotion. The classes used are: ground (walkable), plane (non-walkable, usually vertical), obstacle (non-walkable and not planar), stairs (walkable with caution), foliage (possibly traversable, maybe with a different gait), and sky (neither traversable nor obstacular). These are colored yellow, red, magenta, cyan, green and blue respectively in all examples. This particular choice of classes is somewhat arbitrary – and our algorithm is not specific to this choice of course – but we believe this represents a reasonably minimal set of necessary classes to facilitate locomotion through different environments.

To demonstrate the use of orientation in enhancing segmentation, we developed a relatively simple means of classifying and segmenting images. We segment an image by describing a grid of points with a collection of feature vectors, consisting of visual and pose information. These are used to predict the most likely class for each point with a pre-trained classifier. Since each point is classified independently, this initial segmentation exhibits much noise. To mitigate this, we experiment with a Markov random field (MRF) algorithm to enforce a smoothness constraint across the grid of points; or alternatively a conditional random field (CRF) to create more detailed segmentations. Using any of these approaches we show that fusing visual and orientation information can substantially improve segmentation accuracy over using either alone; and crucially, that orientation information enhances performance beyond using position within the image as a feature.

We also show that classification of lines detected in the image can be enhanced by adding location and orientation features, as well as features encoding properties of the lines themselves (non-visual features are collectively referred to as pose features). Finally, we show that combining the results from point and line classification can improve performance over either in isolation.

The result of our method is a segmentation of the image, comprised of sets of contiguous points with the same classification which, as Fig. 1 shows, divides the image into regions appropriate for a navigation task (here showing MRF segmentation). The basic algorithm does not produce a per-pixel segmentation, due to the resolution of the grid we use, but every pixel in the image is covered, and every pixel is used for the description; the CRF segmentation goes beyond this by giving an individual label to each pixel (see Fig. 12).

4 Data Acquisition

To develop and evaluate the algorithms in this paper, we gathered long video sequences (totalling around 90 minutes of footage) using an IDS uEye USB 2.0

camera[1] fitted with a wide-angle lens (approximately 80° field of view). This provides images at a resolution of 640×480, at a rate of 30 Hz.

Our aim is to use this method to guide humans through outdoor environments. Therefore, all our data were gathered from a camera mounted on the front of a virtual reality headset, worn by a person traversing various urban environments. While walking, the subject saw only the view through the camera. This was done to make the data as close as possible to what would be encountered in a real application, both in terms of the camera being mounted in the same way, and the movements of the head being typical of a human with limited visibility.

The hardware we used for this was the Oculus Rift[2] (Dev. Kit 1), which has a large field of view and sufficiently high framerate (up to 60 Hz). The camera was mounted sideways, so that the images have a portrait orientation – this is because the view for each eye is higher than it is wide. We correct for barrel distortion introduced by the lens to produce an image approximating a pinhole camera, using camera parameters obtained with the OpenCV calibration tool.[3]

To gather orientation information we used the inertial sensor built into the Oculus Rift. This comprises a three-axis accelerometer, gyroscope and magnetometer, which are combined with a sensor fusion algorithm to give estimates of orientation in a world coordinate frame at 1000 Hz. We retrieve the orientation as three Euler angles, and discard the yaw angle (rotation about the vertical axis), since in general this will not have any relationship to semantic aspects of the world. Conversely, pitch and roll are important since they encode the camera pose with respect to the horizon line, and thus whether the camera is looking up/down or is tilted. This has an influence on the likelihood of different classes being observed.

From these videos, a subset of frames are hand picked for labelling, for training or testing. They are manually segmented into disjoint regions, built from straight-line segments. Each region is assigned a ground truth label from our set of classes. This is by nature a subjective task, since image content is often ambiguous, but the labelling is as consistent as possible. Some truly ambiguous regions are not labelled, which are omitted from all training and testing.

Examples of ground truth data can be seen in Fig. 2. The labelling is independent of the points and lines which are later created in the image. We also show ground truth segmentations derived from these, in which a grid of points has been assigned labels according to the the underlying ground truth (where the blockiness due to the grid is clearly visible). This is the best possible segmentation, against which we evaluate our algorithms in Section 6.

5 Classification and Segmentation

In this section we describe the process by which an image is segmented, according to either the visual features, pose features, or combinations thereof; and how

[1] en.ids-imaging.com
[2] www.oculus.com
[3] www.docs.opencv.org

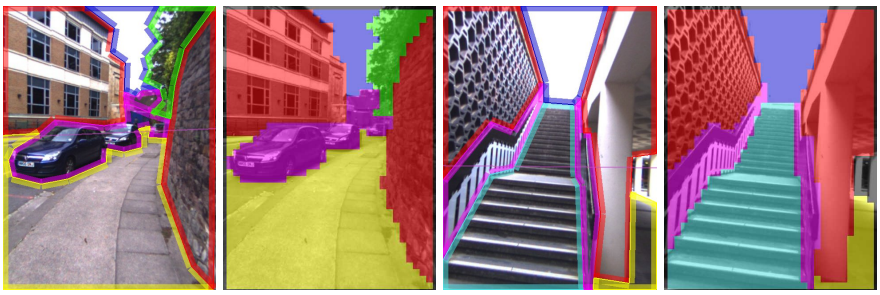


Fig. 2. Example ground truth – the manual labelling of regions (left) and ground truth segmentation (right).

these features are created in regions surrounding grid points, detected lines, or both. An overview of the whole system is presented in Fig. 3.

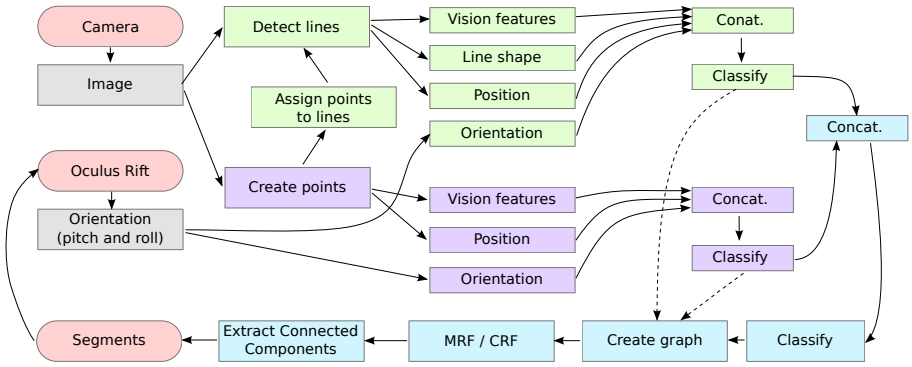


Fig. 3. This block diagram showing how the system as a whole works. The dotted lines show what happens when points or lines are used alone; if both are used, their outputs are concatenated and passed to the meta-classifier.

5.1 Structures

While applying classification at the level of individual pixels is a valid option, this would be very computationally expensive, and the information at one pixel (e.g. its color and location) is unlikely to be sufficiently discriminative. Instead, we use a combination of point and line structures. The points, organized in a grid, are described by features created from their surrounding pixels. Using a regular grid of points also makes segmentation with graphical models more straightforward than using only salient points for example [12]. Lines are detected in the image

use the LSD line segment detector[4] [31] (we discard lines under 6 pixels long and 3 pixels wide (LSD gives a width value for each line) since these are likely to be noise). These are used in order to represent high-frequency image content, and distinctive appearance changes over discontinuities, which would be missed by the smaller and more localized point features. Feature vectors for lines are created from surrounding pixels, extending along their length and covering a region of fixed width on either side.

In order to combine lines with the point-based segmentation, we assign points to lines if they lie within the region enclosed by the line feature (a point may be assigned to multiple lines). It is this assignment of points to lines which later allows line classifications to be transferred to points for segmentation; similarly, the ground truth label of a line is obtained via the points, whose label in turn comes from the marked ground truth regions (thus a line's label vector is the mean label vector of all points inside the area used to describe it).

5.2 Features

The features with which we classify structures in the image are divided into two broad categories: visual features and pose features. The former uses information derived from the image pixels to describe local regions of the image; the latter comprise other information not directly present in the image, but pertaining to properties of image structures or the image as a whole.

Visual Features The visual features we use to describe points are histograms encoding the distribution of gradients and colors in a surrounding square patch. Histograms of gradients describe the local texture, and are built by first convolving the image with gradient filters in the x and y directions, to obtain at each pixel gradient responses g_x and g_y. For each pixel we calculate the gradient angle $\theta = \tan^{-1} \frac{g_y}{g_x}$ and gradient magnitude $m = \sqrt{g_x^2 + g_y^2}$. These are used to build the gradient histogram for a patch by quantising the angles into bins, and weighting the contribution to each bin by their magnitudes. To encode richer structure information we create a separate histogram for each quadrant of the patch and concatenate them together, in the manner of [12].

In addition, color descriptors are created for these patches. These are histograms built in HSV space, which combine a histogram of quantized hue values, weighted by the saturation (since the saturation represents the degree to which the hue is relevant), and an intensity histogram. These are included alongside texture information since color is beneficial when classifying and segmenting images [14, 17].

As mentioned above, we also perform classification on lines detected in the image. In order to create a description better suited to lines, for both gradient and color we create pairs of histograms from the pixels in rectangular regions either side of the line, and concatenate them. Thus, the gradient descriptor has

[4] Code available at www.ipol.im/pub/art/2012/gjmr-lsd

half the dimensionality compared to the point case (which had four quadrants), while the color descriptor is twice as long.

Pose Features The most basic of our pose features is simply the position (of the point, or the line's midpoint) in the image, where we use the x and y coordinates (normalized by image size) directly. This is to represent any dependence on image location which may be exhibited by different classes. Note that the use of location without orientation was not investigated in the original version of this work [13].

The main contribution of this paper is the use of the orientation of the camera as a feature. To obtain this, we use the pitch and roll values from the Oculus Rift IMU, each normalized to the range $[0, 1]$. The orientation features are always combined with image location, since otherwise the orientation feature would be the same for all points in the image: it is the interaction between image position and camera orientation which gives rise to cues of different types of structure at different locations in space.

For the line regions only, we also use a shape descriptor, which comprises simply a line's length, width, and orientation in the image, each appropriately normalized. This is to add extra information – usually at a larger scale – about the scene structure which may be ignored by both the visual and location/orientation features.

5.3 Classification

After extracting features for all points and lines in our training set, each being paired with a ground truth label, we train a set of classifiers. The classifier we use for this work is multivariate Bayesian linear regression [3], chosen because it is both fast to train, and very fast to evaluate for a new input. It is similar to standard regularized linear regression, except that the optimal value for the regularisation parameter can be chosen directly, under the assumption that the data have a Gaussian distribution.

To use it, each class label is represented as a 1-of-K vector (for the $K = 6$ classes), where dimension k is 1 for class k, and zero otherwise. The classifier outputs a K-dimensional vector, which after normalization to sum to 1, is treated as the estimated probability for each class. Prediction is simply a matter of multiplying the feature vector by the $M \times K$ weight matrix (for features of dimensionality M). Rather than the raw feature vector – which would allow for learning only linear combinations of the inputs – we use fourth order polynomial basis functions.

5.4 Combining Information

Different combinations of structures and features lead to different versions of our algorithm. The most basic is using points only (**P**) with visual features (**V**), an algorithm which we will denote **P-V**. Similarly, we can experiment using only location information (which we denote with an '**X**'), orientation information

(**O**), and combinations thereof. For algorithms using line structures (**L**), we can also add the shape feature, denoted by '**S**'.

In order to combine different features together, we simply concatenate them to create one long feature (an alternative method was not found to improve accuracy, c.f. [13]). These combinations will be expressed as **P-VX** for example (points with visual and location features concatenated).

This concatenation is also done for line regions' features (e.g. **L-XOS**, which combines location, orientation and shape features). However, we cannot combine points with lines by simply concatenating their features, because their features are created over different image regions. Instead, we retain separate classifiers for both structures, and combine their outputs by a process known as 'meta-learning' (or sometimes 'stacked generalisation') [2]. The K-dimensional predicted label vectors from the two classifiers are concatenated, and treated as a new feature vector. This is input to a second round of classification, the output of which is another K-d label vector, representing the final probability estimate for each class, thus combining information from the points and lines. We run the classifiers whose outputs we wish to combine (e.g. **P-V** and **L-V**) on the training data to gather example outputs. These predicted label vectors are concatenated and paired with the known ground truth label for each point, so that the meta-classifier can be trained (e.g. resulting in **PL-VO**). Note that this concatenation is done at the points, where the points receive labels from the lines in which they lie. Points not within any line regions simply keep their own predicted label.

5.5 Segmentation

The result of any of the above algorithms is a set of points in the image, each having a predicted label vector, from which we choose the most likely class assignment as the dimension with the highest value. Since each point is classified individually, there is no guarantee that neighbouring points will have similar labels, even if they belong to perceptually similar regions of the image; this is especially true when using line regions, as adjacent points may be assigned to different lines.

Markov random field To address this we formulate the problem as a Markov random field (MRF). This allows us to choose the best label for each point according to its observation (i.e. classification result), while also incorporating a smoothness constraint imposed by its neighbours.

We create a grid graph to represent all the points in the image, by connecting each point to its 4-neighbours. The aim when optimising a MRF is to maximize the probability of the configuration of the field (i.e. an assignment of labels to points); this is equivalent to minimising an energy function over all cliques in the graph [19] (we use up to second-order cliques, i.e. unary and pairwise terms). A configuration of the MRF is represented as $p = (p_1 ... p_N)$, where $p_i \in \mathcal{L}$ is the class assigned to point i of N and $\mathcal{L}$ is the set of possible labels. The goal is to find the optimal configuration p^*, such that $p^* = \mathsf{argmin}_p E(p)$, where $E(p)$ is the posterior energy of the MRF. We define this as:

$$E(p) = \alpha \sum_{i=1}^{N} \psi_d(p_i) + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \psi_s(p_i, p_j) \tag{1}$$

where the first term sums over all points in the graph, and the second sums over all neighbours $\mathcal{N}_i$ for each point i. α is a weight parameter, balancing the effects of the data and the smoothness terms. The unary and pairwise potentials are:

$$\begin{aligned} \psi_d(p_i) &= \|\mathbf{p}_i - \mathbf{c}_i\| \\ \psi_s(p_i, p_j) &= V_{ij} T(p_i \neq p_j) \end{aligned} \tag{2}$$

$\mathbf{p}_i$ denotes the label p_i represented as a 1-of-K vector, and $\mathbf{c}_i$ is the K-d output of the classifier (thus taking into account the predicted probability for all the classes). $T(.)$ is an indicator function, returning 1 iff its argument is true, and V_{ij} is a pairwise interaction term, controlling the degree to which label dissimilarity is penalized at sites i and j. This is set to $V_{ij} = \beta - \mathsf{min}(\beta, |m_i - m_j|)$, where m_i is the median intensity over the patch at point i. This penalizes differences in label more strongly between points with similar appearance, in order to adapt the segmentation to the underlying image contours. We set the parameters to $\alpha = 60$ and $\beta = 90$ empirically based on observations on the training set (note the pixel intensities are in the range $[0, 255]$). We optimize the MRF using graph cuts with alpha-expansion[5] [6]. After optimising the MRF we perform connected-component analysis to recover the segments. Examples of results before and after MRF segmentation can be seen in Fig. 12.

Conditional random field We also describe an alternative way to segment the image, using a conditional random field (CRF). CRFs have an advantage over MRFs in that they model the conditional distribution of the labels with respect to the features, rather than the full joint distribution. This means an accurate conditional model can have a much simpler structure than a fully generative joint model [27]. Recent advances allow for extremely efficient CRF optimisation, so much so that it is now possible to use a fully-connected graph, as opposed to the grid-structured graph described above, connecting every pixel to every other pixel. To do this we use the algorithm of Krähenbühl and and Koltun[6] [17]. For the unary potential at pixel i and label k we use:

$$U_{ik} = \begin{cases} -ln(\mathbf{p}^+_{ki}) \text{ if point } i \text{ has a label} \\ -ln(\frac{1}{K}) \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

where x^+ is the value of x if it is greater than zero (zero otherwise), and $\mathbf{p}_{ik}$ is the kth element of the K-d probability vector predicted at point i. Since the CRF is defined over every pixel, most nodes will not have an initial label. The inference algorithm is otherwise used unchanged, except we doubled the standard deviation of the color-independent term, as we found this to improve performance.

[5] Using the 'gco-v3.0' code at vision.csd.uwo.ca/code
[6] Using code available at www.philkr.net/home/densecrf

6 Results

To evaluate our algorithms, we gathered two datasets as described in Section 4. All data were obtained from the same camera, having a (rotated) resolution of 480×640, and were corrected for barrel distortion due to a wide-angle lens.

The first dataset was designated the training set, and contained 178 manually labelled images. This set was used for cross-validation experiments, to demonstrate the claims made above. The second set of 156 images was the test set, which came from different video sequences recorded in physically distinct locations to the training set. This was done to verify that the algorithms generalize beyond the training set, and to show example images (all examples in the paper come from this set). All our labelled data are available online.[7]

Our algorithm has a large number of parameters which will effect its operation. The most important ones are described here, with typical values given. The grid density (distance between points) was set to a value of 15 pixels (making a grid of approximately 30×40 points), to give a compromise between an overly coarse representation/segmentation, and the quadratic increase in computational time for denser grids. The patches around every pixel, from which visual features are built, were squares of side 20 pixels. The width of line regions was set to 30. The basic gradient histogram was 12-d, making the concatenated quadrant feature on points 48-d; and color histograms had 20 dimensions each for the hue and intensity parts. As described earlier, location and orientation features had two dimensions each, while line shape features have three.

These parameters were set to values which appear sensible or are supported by related literature. However, we make no claim that these were the optimal parameters, and much further tuning could be done, although the best settings would depend on the dataset used. We emphasize that this does not alter the central claims of this work, i.e. that making use of orientation information, using either points or lines, can improve segmentation. All parameters were kept constant across evaluations, so all results are relative.

6.1 Cross-Validation

We begin with results obtained through cross-validation on our training set. This was done by running five independent runs of five-fold cross-validation on the data (to mitigate artefacts due to particular choices of training/test splits). For comparison we use classification accuracy, i.e. the average number of times a point was assigned the correct class. Segmentation was evaluated point-wise, i.e. looking at every point individually, since for the time being we are not concerned with the issue of true segments being wrongly split or merged. We first analyse the performance of the algorithms without the benefits of segmentation: the MRF and CRF are not used here, and we directly used the labels assigned to points by the classifiers.

[7] Our dataset can be found at www.osianh.com/inertial

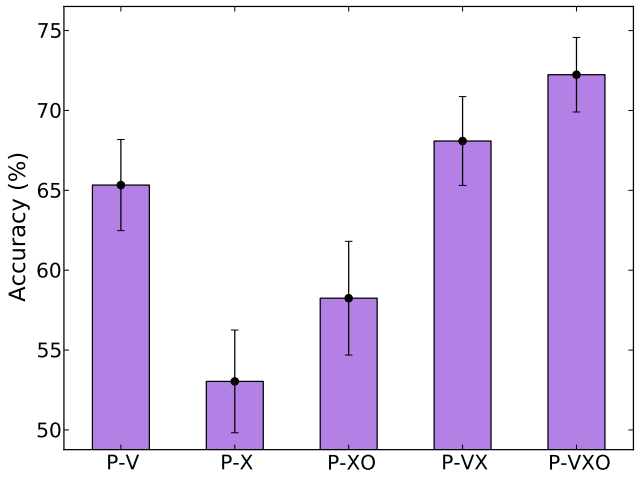


Fig. 4. Adding orientation features to vision features for points. All error bars show a 95% confidence interval.

First, we ran an experiment to evaluate classification using only points, with various combinations of vision and pose features. The results are shown in Fig. 4. The bars indicate the average accuracy over all the runs of cross-validation, and the error bars are drawn to show a 95% confidence interval, based on the average standard error over all runs of cross-validation.

Using vision features alone (**P-V**) provides a reasonable baseline performance. We ran experiments using only the location feature (**P-X**) or location plus orientation (**P-XO**) – as one might expect, these perform much worse than using only vision, since no image information is actually used. Nevertheless, it is encouraging to see that adding orientation already improves the accuracy, and it can be surprising what orientation alone can tell us about what an image is expected to contain, as we will show in the next section.

One of the key results of this paper is that combining vision with orientation information (**P-VXO**) is significantly better than using vision alone. Crucially, we also show that while adding image position as a feature (**P-VX**) does give some improvement (as per [14]), it is the combination of inertial information with image location which gives the largest increase. This was not shown in the original paper [13], but is important in demonstrating that the prior introduced by where the camera is pointing is relevant to classification.

The next experiment was the same as the above, but for line regions instead. Note that these evaluations were done using the points which were assigned labels from classified lines, not on the lines themselves (points not in lines were excluded from the evaluation). As shown in Fig. 5, we tested the use of line location (**L-X**) and shape on their own (**L-S**), in combination (**L-SX**), and combined with orientation (**L-SXO**), again showing the increased performance

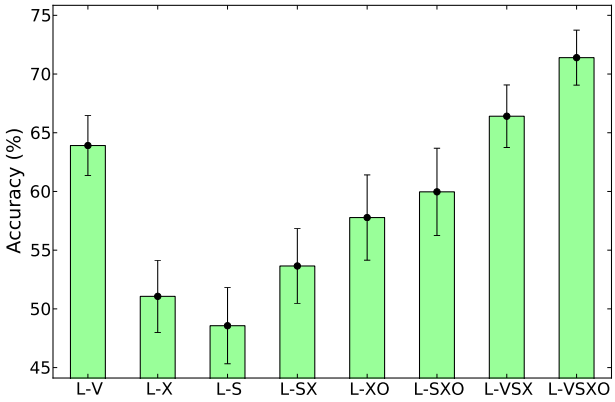


Fig. 5. Using orientation with vision features for lines.

when using orientation. When combining with visual information (**L-VSXO**), we show that this is superior to using visual information alone, or even visual information with shape and location (**L-VSX**), once again demonstrating that the addition of orientation information is of primary importance, and is what allows us to obtain significantly better classification rates.

Finally, we investigated the effect of combining point and line features. Figure 6 first shows both point and lines separately, using only vision features (the same results from the two previous graphs), then the result obtained when combining both point and line classification (**PL-V**). It can be seen that this improves performance compared to using either structure in isolation. We then see the same trend when adding location, orientation and shape information: the graph shows the results of points and lines individually with the full set of features (once again the same as the previous graphs), and finally the result using both structures and all features (**PL-VSXO**). This results in an improved accuracy, suggesting that combining information from multiple types of patch/region is indeed beneficial, albeit by a smaller margin than the above experiments.

In Fig. 7 we show a confusion matrix, obtained as the mean confusion matrix over all runs of cross-validation, using the full algorithm **PL-VSXO**. The diagonal is pleasingly prominent, though there is significant confusion between stairs and ground (when the true class is stairs), which is somewhat unfortunate from a safety point of view. Vertical surfaces also tend to be confused with other obstacles and foliage, which is less of a concern. For our task, ground identification is perhaps the most important criterion, which appears to be the strongest result.

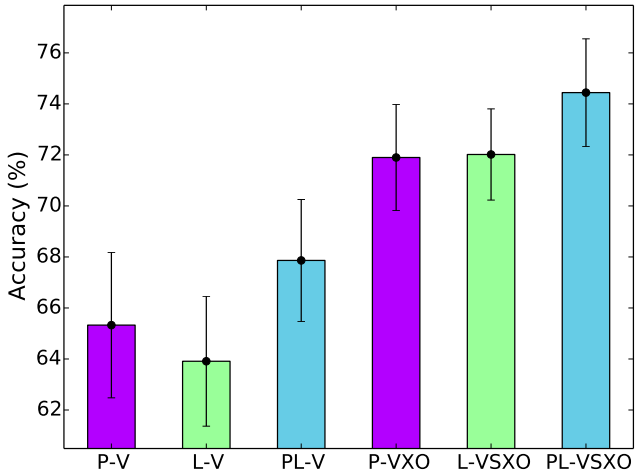


Fig. 6. Combining predictions from both points and lines.

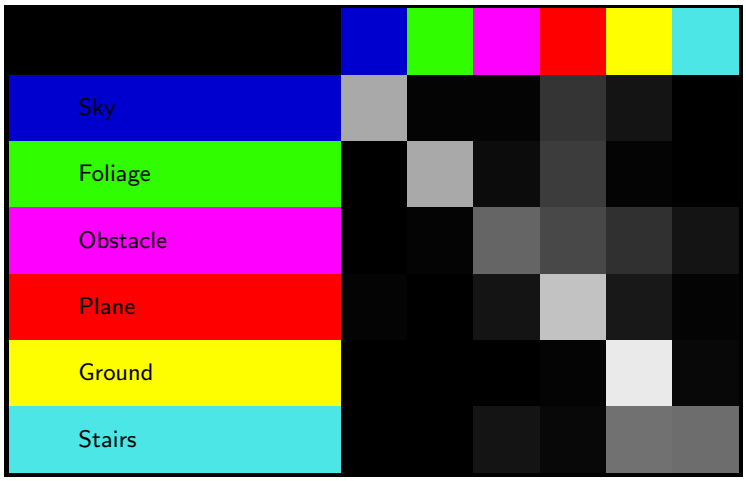


Fig. 7. Confusion matrix over all runs of cross-validation, for complete **PL-VSXO** algorithm. Rows correspond to the true classes, while columns represent the predicted classes. Colors correspond to those used through all segmentation examples.

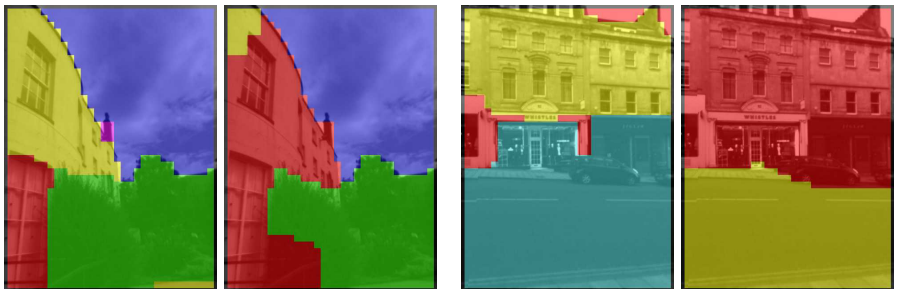
6.2 Independent Data and Examples

After the cross-validation experiments, we trained sets of classifiers corresponding to different variants of the algorithm, using the training set above (plus copies obtained by reflecting across the vertical image axis). We used these to evaluate performance on the independent test set. Results are shown in Table 1. This confirms the important result of the paper: that combining orientation information is beneficial, exhibiting around 10% increase in overall accuracy. Adding line information did confer a further improvement, although this was only slight. We also show results after applying MRF and CRF segmentation, both of which increased accuracy by a few percent.

Algorithm	Accuracy	MRF	CRF
P-V	61.0 %	64.7 %	63.1 %
P-VXO	71.1 %	73.8 %	72.2 %
PL-VSXO	71.5 %	74.6 %	72.5 %

Table 1. Comparison of the different algorithms on independent test data. Using a MRF to smooth away spurious local detections increases accuracy slightly in all cases; a fully connected CRF does not give better performance than the MRF as measured here, but gives more detailed segmentation.

We now show example results taken from the test set, showcasing the differences between the algorithms presented above and demonstrating what they are capable of. In all example images in the paper (except Fig. 12), the MRF segmentation has been run, to remove noise and give a tidier segmentation.



(a) (b)

Fig. 8. Example results, showing segmentation using only vision features (left) and combined with orientation features (right). See color legend in Fig. 7.

First, Fig. 8 shows side by side examples of the basic vision version (**P-V**), and the effect of adding pose information (**P-VXO**). In (a), the building façade is partly mistaken for the ground by the visual features, whereas knowing the

camera is pointing upwards corrects this. In (b) the miss-classification of the road as stairs is also corrected.

In the next example (Fig. 9) we show the effect of adding line classifications to the points-only segmentation, in both cases using all visual and pose features (**P-VXO**, **PL-VSXO**, respectively). These examples show how the information gleaned from the lines can aid segmentation, for example by disambiguating stairs and planes, or finding non-planar objects. However, as our results below will show, lines can sometimes be detrimental.

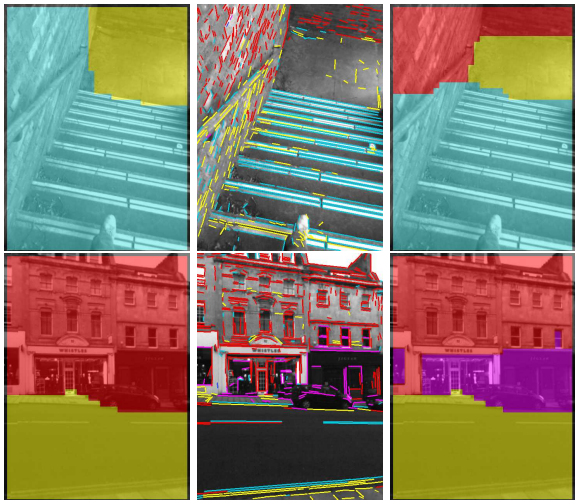
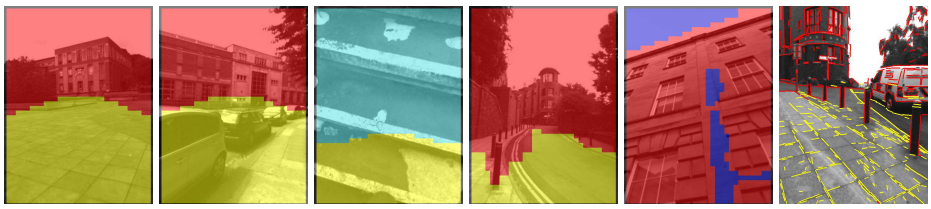


Fig. 9. Examples showing how adding line classifications (centre) in conjunction with point features (**P-VXO**, left) can help improve segmentation (**PL-VSXO**, right).

It is interesting to see what effect the orientation features have, independently of the vision features, so in Fig. 10 we show results generated using **P-XO** and **PL-XOS**, i.e. there are no visual features at all being used in these segmentations (image information is being used only for line detection). Figure 10(a) appears to be correctly segmented, but only because this is a common and rather empty configuration of ground and walls; whereas the cars in (b) are obviously ignored. (c) is interesting since it shows that with the camera looking down at a certain angle, stairs are predicted – in this case correctly. This raises the interesting issue that stairs are predicted here not just because they are likely to be below the viewer, but because the viewer is likely to look downward when walking up stairs. In 10(d) and (e) the use of lines has altered the segmentation, to give the impression it is seeing the bollards and the sky (the points assume there is sky above, but lines even at such a height are rarely labelled as sky in the training set). In (f) the lines themselves are shown, and it can be seen how their orientation in the image has an effect, since the bollards and paving stones are classified differently, despite being at around the same image height.



(a) (b) (c) (d) (e) (f)

Fig. 10. Example segmentations using only orientation information features – points only (a-c) and points with lines (d-f). (f) shows the lines themselves, showing the effect of the lines' orientations within the image, aiding detection of vertical posts.

More examples are shown in Fig. 11. Here, we show the input image for clarity, plus the ground truth segmentation. The contributions from vision (**P-V**), orientation (**P-VXO**), and lines (**L-VSXO**) to the final segmentation (**PL-VSXO**) are shown. Figures 11(a) and 11(b) again show orientation information being used to improve classifications, the latter being an interesting example where adding lines improves segmentation even in the presence of motion blur. Note that the different orientations of the camera, such as in (c) and (d), illustrates why using only position in the image as a prior is inferior.

The segmentation in Fig. 11(e) also benefits from classification of lines along the steps. Similarly in Fig. 11(c) lines help to correctly identify the step-edges, but the step faces are classified as ground. In a way this is correct, since stairs are made up of periodic walkable regions, but this result would be marked mostly incorrect compared to our ground truth, which is labelled at a coarser resolution. This echoes our comment in Section 4 about the world being ambiguous; but also that some regions may belong to multiple classes simultaneously at different scales.

The example in 11(g) also shows orientation information being used to correctly identify the non-ground surface; however, the addition of lines in this case degrades the result. The final two rows show examples where our augmented algorithms fail to provide any benefit. In 11(h), the initial **P-V** segmentation is correct, and is unchanged by the addition of orientation or lines (of course, if we could achieve perfect segmentation, no amount of prior knowledge would help). On the other hand, this illustrates why it is so important that orientation does not impose a hard constraint on surface identity: even when orientation features are added, the grass (foliage class) remains. In 11(i), none of the versions of the algorithm are able to detect either the ground plane or the foliage, perhaps due to the lower level of illumination.

Our implementation, consisting of single-threaded C++ code running on a desktop PC (Intel i5, 2.40 GHz), processes one image in around 0.3 seconds on average (including MRF optimisation). This is below the camera rate, but fast enough for real-time use when run in a separate thread; further improvements

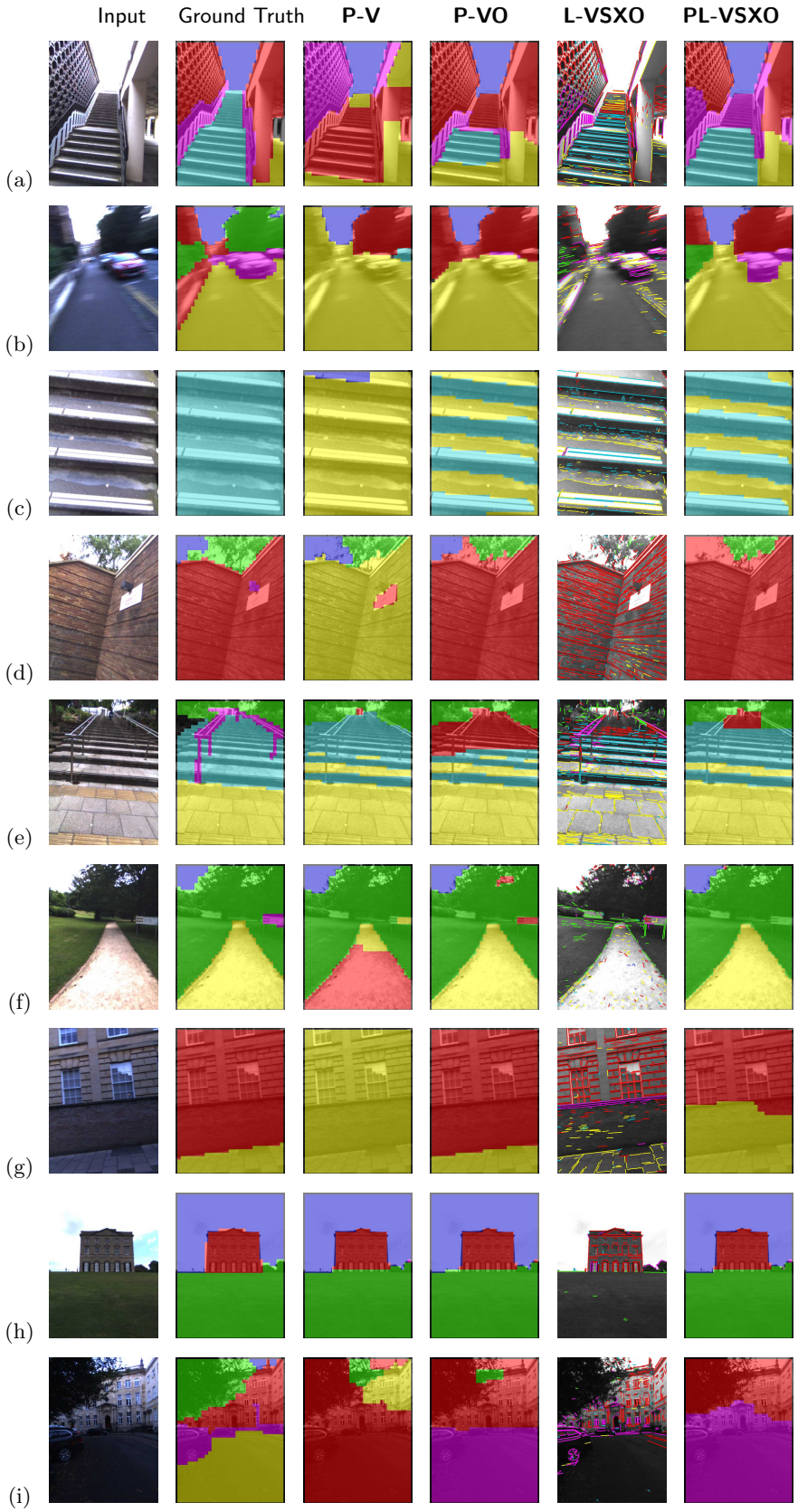


Fig. 11. Example results of the various algorithms. After the input and ground truth, we show the baseline result, of points with only vision features (**P-V**), followed by adding orientation information (**P-VXO**). Detected and vision-classified lines are shown, before the final result, combining everything (**PL-VSXO**).

could be made by parallelising the code or using a GPU. Videos of our code running can be seen on our website.[8]

6.3 Detailed Segmentation

Finally, we show results of the algorithm when segmenting using the CRF (see section 5.5). This is much slower than the MRF (taking over 1s per image), but since it uses a fully-connected graph of every pixel in the image it results in much more detailed segmentations. The fully-connected nature of the graph means that relationships between distant regions of the image can be taken into account, which can help improve the robustness of the segmentation; however, this also means that classifications in one region can be influenced by those in another, so that a region's label may change as different parts of the scene come into view. As Table 1 shows, the CRF gives some improvement on the raw output, but does not (in its current configuration) out-perform the MRF (note that we are only evaluating using the points, as before, so the evaluation cannot benefit from the improved resolution). Nevertheless, as the examples in Fig. 12 show, the dense CRF can give a significantly more precise and detailed segmentation of the image. It helps to more clearly delineate small objects such as bollards (Fig. 12(a)) and complex boundaries like trees (c,d), although it can also introduce some misclassifications (e). Videos of this being run in a threaded real-time system are also available on our website.

7 Conclusion

We have presented a way of combining information about the real-world orientation of a camera, obtained through inertial measurements, with more traditional vision features, for an image segmentation algorithm. This focused on our example application of scene segmentation for locomotion in outdoor environments, but we would expect the results to be applicable to other types of classification, segmentation, scene understanding and image parsing tasks where the orientation of the camera is likely to effect the image content. We have also shown that adding orientation information is beneficial for line regions; and that combining points and lines in a similar manner can lead to some further improvement.

Our experiments used a comparatively basic design of segmentation algorithm to highlight the effect of using extra prior information. While we have also experimented with using more advanced segmentation techniques, the CRF segmentation took as input our classified image. An interesting avenue of further research would be to combine the orientation prior with the visual features within the graphical model framework [27], to make use of the graph structure at the classification stage too.

Other future work will look at ways of using this method with other sources of information, for example making use of temporal information to enforce consistency across frames, or to combine with depth and 3D data.

[8] Videos available at www.osianh.com/inertial

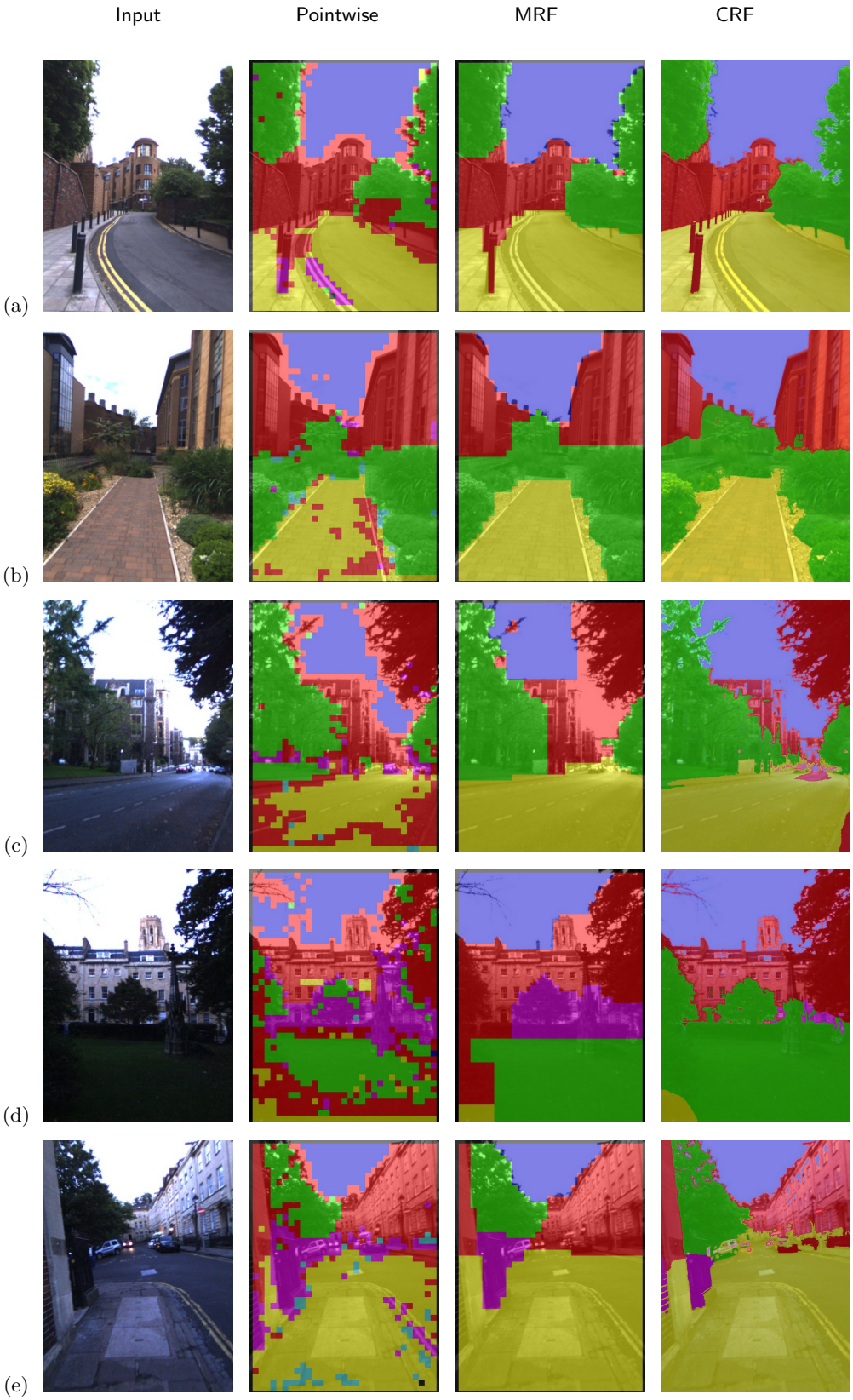


Fig. 12. This shows the result of the pointwise classification before any segmentation or smoothing (second column). The MRF reduces noise by imposing a smoothness constraint, and groups together points with the same class (third column). We also show results using a dense fully-connected CRF (fourth column), which assigns a label to every pixel using the pointwise result as input.

Acknowledgments. This work was funded by the UK Engineering and Physical Sciences Research Council (EP/J012025/1). The authors would like to thank Austin Gregg-Smith and Geoffrey Daniels for help with hardware and data, and Adeline Paiement for all the enlightening discussions.

References

1. Angelaki, D.E., Cullen, K.E.: Vestibular System: The Many Facets of a Multimodal Sense. Annual Review of Neuroscience. 31, 125–150 (2008)
2. Bi, Y., Guan, J. and Bell, D.: The combination of multiple classifiers using an evidential reasoning approach. Artificial Intelligence. 172 (15), 1731–1751 (2008)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
4. Brandt, T., Bartenstein, P., Janek, A., Dieterich, M.: Reciprocal inhibitory visual-vestibular interaction. Brain. 121 (9), 1749-1758 (1998)
5. Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S. and Bradski, G.R.: Self-supervised Monocular Road Detection in Desert Terrain. In: Robotics Science and Systems (2006)
6. Delong, A., Osokin, A., and Isack, H.N, and Boykov, Y.,: Fast approximate energy minimization with label costs. Int. J. Computer Vision. 96(1), 1–27 (2012)
7. Deshpande, N., Patla, A.E.: Visual–vestibular interaction during goal directed locomotion: effects of aging and blurring vision. Experimental brain research. 176 (1), 43–53 (2007)
8. DeSouza, G.N., Kak, A.C.: IVision for mobile robot navigation: A survey. IEEE Trans. Pattern Analysis and Machine Intelligence. 24 (2), 237–267 (2002)
9. Domke, J.: Learning graphical model parameters with approximate marginal inference. IEEE Trans. Pattern Analysis and Machine Intelligence. 35 (10), 2454 (2013)
10. Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: IEEE Int. Conf. Computer Vision (2009)
11. Gupta, S., Arbeláez, P., Girshick, R., Malik, J.: Indoor Scene Understanding with RGB-D Images: Bottom-up Segmentation, Object Detection and Semantic Segmentation. Int. J. Computer Vision, 1–17 (2014)
12. Haines, O., Calway, A.: Recognising Planes in a Single Image. IEEE Trans. Pattern Analysis and Machine Intelligence. 37 (9), 1849–1861 (2014)
13. Haines, O., Bull, D., Burn, J.F.:Using Inertial Data to Enhance Image Segmentation. In: Int. Conf. Computer Vision Theory and Applications (2015)
14. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. Int. J. Computer Vision. 1 (75), 151–172 (2007)
15. Joshi, N., Kang, S.B., Zitnick, C.L., Szeliski, R.: Image deblurring using inertial measurement sensors. ACM Trans. Graphics. 29(4), p. 30 (2010)
16. Kleiner, A., Dornhege, C.: Real-time localization and elevation mapping within urban search and rescue scenarios. J. Field Robotics. 24 (8-9), 723–745 (2007)
17. Krähenbühl, P. and Koltun, V.: Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In: Advances in Neural Information Processing Systems (2011)
18. Kundu, A., Li, Y., Dellaert, F., Li, F., Rehg, J.M.: Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In: European Conf. Computer Vision (2014)
19. Li, S.Z.: Markov Random Field Modeling in Image Analysis. Springer-Verlag (2009)

20. Lorch, O., Albert, A., Denk, J., Gerecke, M. , Cupec, R., Seara, J.F., Gerth, W., Schmidt, G.: Experiments in vision-guided biped walking. In: IEEE Int. Conf. Intelligent Robots and Systems (2002)
21. Maimone, M., Cheng, Y., Matthies, L..: Two years of visual odometry on the mars exploration rovers. J. Field Robotics. 24 (3), 169–186 (2007)
22. Nützi, G., Weiss, S., Scaramuzza, D., Siegwart, R.: Fusion of IMU and vision for absolute scale estimation in monocular SLAM. Journal of Intelligent and Robotic Systems. 61, 287–299 (2011)
23. Patla, A.E.: Understanding the roles of vision in the control of human locomotion. Gait & Posture. 1 (5), 54–69 (1997)
24. Piniés, P., Lupton, T., Sukkarieh, S.,Tardós, J.D. Inertial aiding of inverse depth SLAM using a monocular camera. In: Int. Conf. Robotics and Automation. (2007)
25. Sadhukhan, D., Moore, C., Collins E.: Terrain estimation using internal sensors. In: Int. Conf. Robotics and Applications (2004)
26. Gould, S., Fulton, R., Koller, D.: Combining appearance and structure from motion features for road scene understanding. In: British Machine Vision Conf. (2009)
27. Sutton, C., McCallum, A.: An introduction to conditional random fields for relational learning. Introduction to Statistical Relational Learning. 93–128 (2006)
28. Tapu, R., Mocanu, B., Zaharia, T.: A computer vision system that ensure the autonomous navigation of blind people. In: Conf. E-Health and Bioengineerin (2013)
29. Vidal, P.P., Degallaix, L., Josset, P., Gasc, J.P., Cullen, K. E.: Postural and locomotor control in normal and vestibularly deficient mice. The Journal of Physiology. 559 (2), 625638 (2004)
30. Virre, E.: Virtual reality and the vestibular apparatus. Engineering in Medicine and Biology Magazine. 15 (2), 41–43 (1996)
31. Von Gioi, R.G., Jakubowicz, J., Morel, J., Randall, G.: LSD: A fast line segment detector with a false detection control. IEEE Trans. Pattern Analysis and Machine Intelligence. 32 (4), 722–732 (2010)