A neural implementation of the Hough transform and the advantages of explaining away

M.W. Spratling

PII: S0262-8856(16)30077-4
DOI: doi: 10.1016/j.imavis.2016.05.001
Reference: IMAVIS 3502

To appear in: *Image and Vision Computing*

Received date: 15 September 2015
Accepted date: 2 May 2016

A neural implementation of the Hough Transform and the advantages of explaining away

M. W. Spratling

King's College London, Department of Informatics, London. UK. michael.spratling@kcl.ac.uk

Correspondence should be addressed to: M. W. Spratling
Department of Informatics
King's College London
Strand
London WC2R 2LS
UK

email: michael.spratling@kcl.ac.uk
telephone: +44 20 7848 2027

Submitted: Tuesday 15th September, 2015

Abstract

The Hough Transform (HT) is widely used for feature extraction and object detection. However, during the HT individual image elements vote for many possible parameter values. This results in a dense accumulator array and problems identifying the parameter values that correspond to image features. This article proposes a new method for implementing the voting process in the HT. This method employs a competitive neural network algorithm to perform a form of probabilistic inference known as "explaining away". This results in a sparse accumulator array in which the parameter values of image features can be more accurately identified. The proposed method is initially demonstrated using the simple, prototypical, task of straight line detection in synthetic images. In this task it is shown to more accurately identify straight lines, and the parameter of those lines, compared to the standard Hough voting process. The proposed method is further assessed using a version of the implicit shape model (ISM) algorithm applied to car detection in natural images. In this application it is shown to more accurately identify cars, compared to using the standard Hough voting process in the same algorithm, and compared to the original ISM algorithm.

Keywords Hough Transform; Generalised Hough Transform; Implicit Shape Model; feature extraction; object detection; explaining away; neural networks.

1 Introduction

The Hough Transform (HT) was introduced more than half a century ago (Hart, 2009; Hough, 1962) but remains a popular method in image processing and computer vision having been extended to detect features, objects, or shapes of any type (Ballard, 1981; Beinglass and Wolfson, 1991; Davies, 1988; Duda and Hart, 1972; Gall et al., 2011; Leibe et al., 2004; Maji and Malik, 2009; Samal and Edwards, 1997; Yao et al., 2010). In general terms, the HT is a process of summing up evidence (accumulating votes) for a shape from multiple, local, image elements that could be constituent parts of that shape. Evidence is summed in an accumulator array in which different cells represent different combinations of parameters (such as the location, scale, orientation, *etc.*) for possible instances of the shape[a]. Following this voting process, cells with high values will correspond to those instances of the shape for which there is most evidence. Shape detection thus involves finding the peaks in the accumulator array.

The HT suffers from several well known problems including spurious peaks and quantization effects. The former problem refers to the difficulty in identifying cells in the accumulator array that correspond to peaks caused by the presence of a shape-instance, and distinguishing these from other cells that have high values caused by the spurious addition of votes from image elements that do not form a single shape-instance. The latter problem occurs when different constituent parts of the same shape-instance fail to place votes in the same cell of the accumulator array due to the true parameters of the shape-instance not corresponding to the parameters values represented by the cells in the accumulator array.

This article proposes a method of improving the HT, and of solving the problems of spurious peaks and quantization effects, by changing the way in which the votes are cast. Specifically, it is proposed that the HT can be implemented using a neural network, such that each element in the accumulator array is replaced by a neuron. The activity of each neuron corresponds to the evidence accumulated for a particular combination of parameters (*i.e.*, for a shape-instance). For example, in the traditional application of the HT to straight line detection, the accumulator array consists of a 2-dimensional array

[a]The word “shape” will be used to refer to the type of shape that the HT is being used to detect (*e.g.*, lines or squares or cars), and where it is necessary to make a distinction the phrase “shape-instance” will be used to refer to an instance of the shape with specific parameters (*e.g.*, a car at a particular location and scale).

each elements of which corresponds to a possible position and orientation of a line in the image. In the proposed method, this 2-dimensional accumulator array would be replaced by a vector of neuron activation values, $\mathbf{y}$. The weights of each neuron define how image elements vote for the shape-instance represented by that neuron. For example, in straight line detection the weights $\mathbf{w}_j$ received by neuron j would be zero from each pixel of the image except those pixels lying along the particular straight line represented by that neuron. The weights for all the neurons would be represented by a matrix, $\mathbf{W}$, where each row corresponds to the weights received by a single neuron. The inputs to the neural network would correspond to the image elements, or object parts, that vote for shape-instances. For example, in straight line detection the input is a (typically binary) image where only putative edge pixels have nonzero values. This edge image would be replaced in the proposed method by a vector, $\mathbf{x}$ containing the same values. The voting process performed by the HT can then be implemented in a neural network as follows:

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

Using a neural network where the outputs are a linear combination of the inputs, as shown in the preceding equation, will produce results identical to those produced using the standard Hough voting method. However, implementing the HT as a neural network suggests other ways of calculating $\mathbf{y}$, and hence, other ways to carry out the voting process. Specifically, by implementing the voting process using a neural network in which the neurons compete to be active it is possible to produce an accumulator array in which activity, $\mathbf{y}$, is sparse with only the neurons that best explain the image elements being strongly active. The particular method of competition used in this article performs a form of probabilistic inference known as "explaining away" (Kersten et al., 2004; Lochmann and Deneve, 2011; Lochmann et al., 2012; Spratling, 2012b; Spratling et al., 2009). Neurons represent hypotheses about the shape-instances present in the image, and the input to the network represents sensory evidence for these different hypotheses. Neurons compete to explain the evidence. Neurons that are best supported by the evidence have high activity which inhibits other neurons from receiving activation from the same evidence. Hence, if one hypothesis explains the presence of image elements then those elements are prevented from supporting alternative hypotheses (*i.e.*, voting for other shape-instances). This solves the problem of spurious peaks. Additionally, in neural networks it common to represent a stimulus by the joint activity of a number of neurons. Such a "population coding" approach allows a continuous space to be represented using a finite number of neurons. Here, this method is used to interpolate between the parameter values represented by individual neurons, and hence, to overcome the problem of quantization effects. This population decoding approach also relies on explaining away. Explaining away produces a sparse accumulator array, and hence, results in isolated peaks of activity that can be easily distinguished from each other and decoded separately.

Explaining away has previously been proposed as a method of improving the Hough voting process (Barinova et al., 2012; Gerig, 1987; Woodford et al., 2014). However, these methods have typically relied on manipulating the accumulator array after the votes have been cast, rather than (as proposed here), changing the way the votes are cast. For example, Barinova et al. (2012) proposed a probabilistic voting mechanism that performs explaining away. However, in practice this method is implemented by performing standard Hough voting, and then rejecting peaks whose evidence can be explained by other peaks. A similar approach is used in the implicit shape model (ISM; Leibe et al., 2004, 2008) which uses a "Minimum Description Length" (MDL) criteria to reject peaks where the votes come from image elements which have voted for other peaks that are believed to be the true ones. Such post hoc analysis of the accumulator array has also been used to improve the HT when it is applied to the specific task of straight line detection (*e.g.*, Furukawa and Shinagawa, 2003; Ji et al., 2011; Xu et al., 2015). These techniques analyse the distribution of the votes in the accumulator array in order to more accurately locate peaks that correspond to lines in the original image. Previous proposals to solve the problem of quantization effects have used a continuous (rather than a discrete) space in which to accumulate votes, and the peaks in this continuous parameter space are then found using a mean-shift clustering algorithm (Leibe et al., 2004, 2008). Alternatively, it is possible to use a discretized parameter space but to adaptively refine the resolution of this space (Illingworth and Kittler, 1987).

To demonstrate the proposed method it is used to implement the HT for straight lines and to implement the generalised HT applied to object detection in natural images (as in the ISM). The results show that implementing the HT using a neural network performing explaining away results in a large improvement in detection accuracy in both these applications compared to the standard method of Hough voting. Furthermore, the results for object detection in natural images are competitive with state-of-the-art techniques.

2 Methods

2.1 The PC/BC-DIM algorithm

The neural network implementation of the HT proposed in this article is realized using the PC/BC-DIM algorithm (Spratling, 2008, 2010, 2012a,b, 2013, 2014a; Spratling et al., 2009), which is implemented using the following equations:

$$\mathbf{e} = \mathbf{x} \oslash (\epsilon_2 + \mathbf{V}\mathbf{y}) \quad (1)$$

$$\mathbf{y} \leftarrow (\epsilon_1 + \mathbf{y}) \otimes \mathbf{W}\mathbf{e} \quad (2)$$

Where $\mathbf{y}$ is a (n by 1) vector of "prediction neuron" activations which represent the values in the accumulator array; $\mathbf{x}$ is a (m by 1) vector of inputs representing the image elements that vote for shape-instances, $\mathbf{e}$ is a (m by 1) vector of "error neuron" activations which mediate the competition between the prediction neurons; $\mathbf{W}$ is a (n by m) matrix of weight values defining how each image element votes for each shape-instance; $\mathbf{V}$ is a (m by n) matrix of weight values ($\mathbf{V}$ is equal to the transpose of $\mathbf{W}$ but each column is normalised to have a maximum value of one); ϵ_1 and ϵ_2 are parameters; and $\oslash$ and $\otimes$ indicate element-wise division and multiplication respectively. Note that, in many applications $\mathbf{x}$ is sparse, as there are only a limited number of locations in any image where voting elements will be present. Values of $\mathbf{x}$ that are equal to zero have no influence on the resulting values of $\mathbf{y}$ which are calculated via equations 1 and 2. This means that to reduce the computational burden, columns of $\mathbf{W}$ (and rows of $\mathbf{V}$) which correspond to voting elements that are not present in the current image, can be removed. Rows of $\mathbf{W}$ (and columns of $\mathbf{V}$) which contain only zero weight values (corresponding to cells in the accumulator array for which no image element can vote), can also be removed to improve computation speed.

To perform the HT, the inputs $\mathbf{x}$ were presented to the network. Initially, the values of $\mathbf{y}$ were all set to zero, and equations 1 and 2 were then iteratively updated with the new values of $\mathbf{y}$ calculated by equation 2 substituted into equations 1 and 2 to recursively calculate the neural activations. This iterative process was terminated after 50 iterations which was sufficient for the network to reach a steady-state. The values of $\mathbf{y}$ produced after 50 iterations were used as the outcome of the proposed neural Hough voting process.

For all the experiments described in this paper ϵ_1 and ϵ_2 were given the values 1×10^{-6} and 1×10^{-3} respectively. Parameter ϵ_1 prevents prediction neurons becoming permanently non-responsive. It also sets each prediction neuron's baseline activity rate and controls the rate at which its activity increases when new evidence appears at the input to the neural network. Parameter ϵ_2 prevents division-by-zero errors and determines the minimum strength that an input is required to have in order to effect prediction neuron response. As in all previous work with PC/BC-DIM, these parameters have been given small values compared to typical values of $\mathbf{y}$ and $\mathbf{x}$, and hence, have negligible effects on the steady-state activity of the network. Here, as in previous work with PC/BC-DIM only non-negative weights, inputs, and activations are used.

Prediction neurons that share inputs (*i.e.*, that receive votes from the same image parts) will inhibit each other's inputs. This generates a form of competition between the prediction neurons, such that each neuron effectively tries to block other prediction neurons from responding to the pattern of inputs (*i.e.*, the configuration of voting elements) which it represents. This mechanism of competition is implemented using a form of divisive input inhibition (equation 1), in contrast to similar mechanisms of competition which have been implemented using subtractive inhibition (Harpur and Prager, 1996, 1994;

Rao and Ballard, 1999; Spratling, 1999; Spratling and Johnson, 2002, 2003). However, the divisive method is preferred here as it typically converges to a solution more quickly, and the solution is sparser. The competition between the prediction neurons in a PC/BC-DIM network performs explaining away (Kersten et al., 2004; Lochmann and Deneve, 2011; Lochmann et al., 2012; Spratling, 2012b; Spratling et al., 2009). If a prediction neuron wins the competition to represent a particular pattern of inputs, then it inhibits other neurons from responding to those same inputs. Hence, if one hypothesis (*i.e.*, shape-instance) explains part of the evidence (*i.e.*, a sub-set of image parts), then support from this evidence for alternative hypotheses is reduced, or explained away.

The vector of prediction neuron activations, $\mathbf{y}$, can be rearranged to form an array where the cells represent the same parameter values as the cells in the accumulator array generated by the standard Hough voting method. However, unlike the standard Hough voting scheme, the accumulator array produced by PC/BC-DIM will be sparse, and contain isolated regions of activity surrounded by regions in which the activity is close to zero. To locate each of these regions of activity, cells were identified that formed a contiguous neighbourhood in which each cell had an activity of more than 0.001, and which was completely surrounded by cells with a value of 0.001 or less. The parameter values represented by each region of non-zero activity were then determined using population vector decoding (Georgopoulos et al., 1986). This simply calculates the weighted average of the parameter values represented by each cell in the region. Specifically, if cell i in the region has activity value $\mathbf{y}_i$ and represents the parameter value p_i, then the parameter value represented by the region was calculated as:

$$\frac{\sum_i y_i p_i}{\sum_i y_i}$$

This population decoding was calculated separately for each parameter (*i.e.*, along each dimension of the accumulator array). The total sum of the response in each region (*i.e.*, $\sum_i y_i$) was also recorded. This was used to indicate the total votes accumulated for the corresponding parameter values (*i.e.*, for that shape-instance).

2.2 The PC/BC-DIM implementation of the HT for straight line detection

In this application, the voting elements were edge pixels and cells in the accumulator array represented the radius (ρ) and orientation (θ) of the normal to possible straight lines (Duda and Hart, 1972). The input was an a by b pixel binary image, where each non-zero pixel indicated the presence an edge at that location. This was rearranged to form a ab by 1 vector, $\mathbf{x}$. The output $\mathbf{y}$ was a pq by 1 vector containing the votes cast in each cell of the p by q accumulator array. Each row of $\mathbf{W}$ was a binary vector with nonzero values corresponding to each pixel lying on the straight line represented by the corresponding cell in the accumulator array. Equivalently, each column of $\mathbf{W}$ was a binary vector with nonzero values corresponding to each cell in the accumulator array for which one pixel could vote.

2.3 The PC/BC-DIM implementation of the ISM for object detection

In this application, the voting elements were grayscale image patches that were matched to patches in an appearance codebook. The accumulator array was either two-dimensional, representing the x- and y-coordinates of the centre of the object, or was three-dimensional representing the centre coordinates and scale of the object. When applied to locating objects at a single scale (using a two-dimensional accumulator array), the following procedure was used.

To define the appearance codebook, image patches were extracted from around keypoints in each training image. Keypoints were identified using both the Harris and SIFT detectors, and image patches were 15 by 15 pixels. Patches that fell off the edge of the image were discarded. The remaining patches were clustered using the hierarchical agglomerative clustering algorithm, with zero-mean normalised cross correlation (ZMNCC)[b] between the most different members of each cluster as the measure of

[b]Also known as the sample Pearson correlation coefficient.

similarity. Clustering was terminated once the ZMNCC between all clusters was less than 0.4. Those clusters with fewer than 13 members were discarded. The arithmetic mean of the patches forming the remaining clusters were used as the appearance codebook. This process was performed twice, once for training images containing the object (to produce the positive codebook), and once for training images that did not contain the object (to produce the negative codebook). Clusters in the positive codebook that had a ZMNCC with any cluster in the negative codebook that exceeded 0.93 were discarded.

For each member in the positive codebook the distribution of votes, relative to the centre of the image patch, were defined as follows. For each positive training image, keypoints were located using the Harris and SIFT detectors. For the patch of image surrounding each keypoint the best matching codebook entry from both codebooks was found (with matching similarity measured using the ZMNCC). If the best matching patch was from the positive codebook, the position of the centre of the object relative to the keypoint location was recorded in the "voting mask" associated with that codebook entry. Finally, each voting mask was smoothed using a two-dimensional circular symmetric Gaussian function with a sigma of two pixels.

To locate the object in a new image, keypoints were located in the new image using the Harris and SIFT detectors. For the patch of image around each keypoint the best matching codebook entry from both codebooks was found (using the ZMNCC as the similarity metric). If the best matching patch was from the positive codebook, votes were recorded relative to the location of the keypoint using the voting mask corresponding to the matched codebook entry. For the standard Hough voting scheme, the votes can be cast by defining an initially blank "matching map" for each positive codebook entry. Each time the codebook entry is matched to a keypoint, the matching map for that codebook entry is updated by changing the value at the coordinates of the keypoint to be equal to the ZMNCC between the image patch and the codebook entry. Once all matching maps have been updated for all matched keypoints, votes can be cast by convolving the voting masks with the corresponding matching maps, and summing these results across all positive codebook entries. The PC/BC-DIM algorithm can also be implemented using convolution (Spratling, 2014b). However, here votes were cast by concatenating the vectorized matching maps to form the vector $\mathbf{x}$, and defining the weight matrix, $\mathbf{W}$, so that each column contained the a concatenation of the vectorized voting masks for the corresponding elements of $\mathbf{x}$.

When applied to locating objects at a multiple scales (using a three-dimensional accumulator array), the same procedure was used except that separate positive codebooks were trained for each scale, images patches were 17 by 17 pixels in size (due to objects being, on average, at a larger scale than was the case for the single scale task), and the voting masks were smoothed using a three-dimensional Gaussian function with a sigma of two in the x- and y-directions, and one in the scale dimension.

Code

Open-source software, written in MATLAB, which performs all the experiments described in this article is available for download from: `http://www.corinet.org/mike/Code/pcbc_hough.zip`.

3 Results

To quantitatively assess the performance of the proposed algorithm the procedures advocated in Agarwal and Roth (2002) were followed. Specifically, given an accumulator array the peaks exceeding a given threshold level were extracted and the parameter values represented by these peaks were compared to ground-truth data. For standard Hough voting, the threshold was applied to the cell values after non-maximum suppression. For the PC/BC-DIM method of voting, the threshold was applied to the sum of the votes corresponding to each isolated peak in the accumulator array (see section 2.1). If the parameter values were sufficiently close to the ground-truth (see the following sections for the exact criteria used in each task), this was counted as a true-positive. If multiple peaks in the accumulator array corresponded to the same ground-truth parameters, only one match was counted as a true-positive, and the rest were counted as false-positives. All other peaks in the accumulator array that failed to match the ground-truth

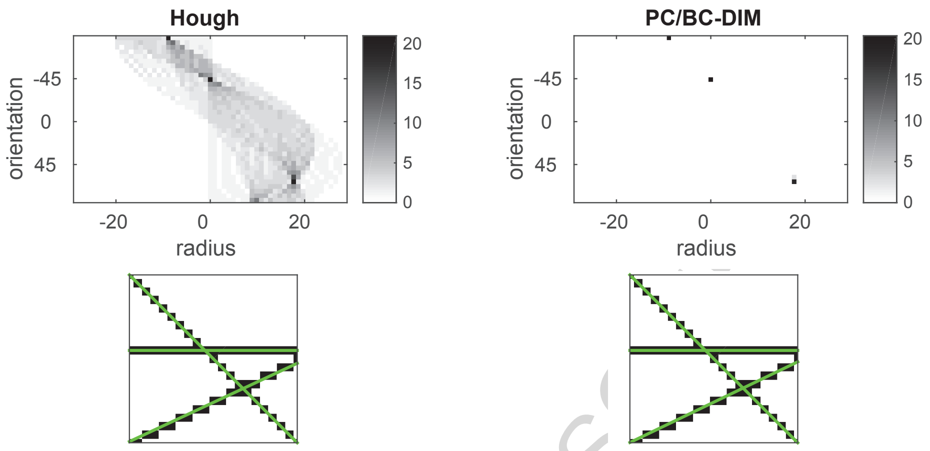

Figure 1: Results for a simple synthetic image containing three straight lines, when using a θ resolution of 5 and a ρ resolution of 1. The top row shows the accumulator arrays produced by standard Hough voting, and by the PC/BC-DIM method of voting. The bottom row shows the original binary images (black pixels are edge pixels), and (in green) the lines detected by thresholding the corresponding accumulator array at a value of 12. Standard Hough voting finds three lines with estimated parameters of (-90,-9), (-45,0), (65,18) that correspond to the lines in the image. PC/BC-DIM finds three lines with estimated parameters of (-90,-9), (-45,0), (64.3,18) that each correspond to a line in the image.

data were also counted as false-positives. Ground-truth parameters for which there was no corresponding values found by the HT were counted as false-negatives. The total number of true-positives (TP), the number of false-positives (FP), and the number of false-negatives (FN) were recorded over many test images, and were used to calculate recall ($\frac{TP}{TP+FN}$) and precision ($\frac{TP}{TP+FP}$). By varying the threshold applied to the accumulator array, precision recall curves were plotted to show how detection accuracy varied with threshold. To summarise performance, the f-score ($= \frac{2.recall.precision}{recall+precision} = \frac{2TP}{2TP+FP+FN}$) which measures the trade-off between precision and recall, was calculated at the threshold that gave the highest value. In addition, to allow comparison with previously published results, the equal error rate (EER) was also found. This is the percentage error when the threshold is set such that the number of false-positives equals the number of false-negatives.

3.1 Straight line detection

To illustrate of the behaviour of the proposed algorithm it was initially applied to the task of detecting straight lines. The effects of explaining-away produced by the proposed voting method can be seen by comparing the accumulator arrays generated by PC/BC-DIM with those generated by standard Hough voting. The qualitative performance of each method is assessed by showing the straight lines found by each method (at a single threshold) projected back onto the original image. Quantitative assessment of performance is given later in this section.

Figure 1 shows the results for a simple synthetic image containing three straight lines with parameter values (θ, ρ) of (-90,-9), (-45,0) and (63.4,17.9). The standard Hough voting method results in a dense accumulator array. Following the application of non-maximum suppression, it is possible to find a threshold that accepts only those cells that correspond to lines in the image. In this particular example, the cells that correspond to lines have values of 21 or 20. However, there are numerous other cells with relatively high values in the accumulator array after non-maximum suppression (for example, there are a total of 7 cells with values greater than 8 and 23 cells with values greater than 5). In contrast, the PC/BC-DIM method of voting results in a sparse accumulator array with exactly three clearly identifiable peaks.

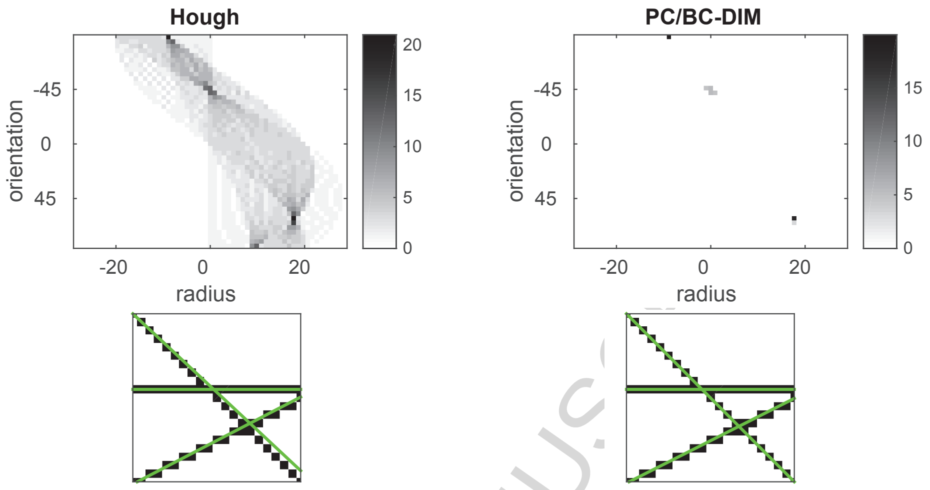

Figure 2: Results for the same image used in Fig. 1 when using a θ resolution of 3.9 and a ρ resolution of 1. The format is the same as, and described in the caption to, Fig. 1. Using a threshold of 12, standard Hough voting finds three lines with estimated parameters of (-90,-9), (-47,0), (62.61,18) that correspond to the lines in the image, but the parameters of one line are poorly estimated. PC/BC-DIM finds three lines with estimated parameters of (-90,-9), (-45,0), (63.2,18) that are all close to the true parameters of the lines in the image.

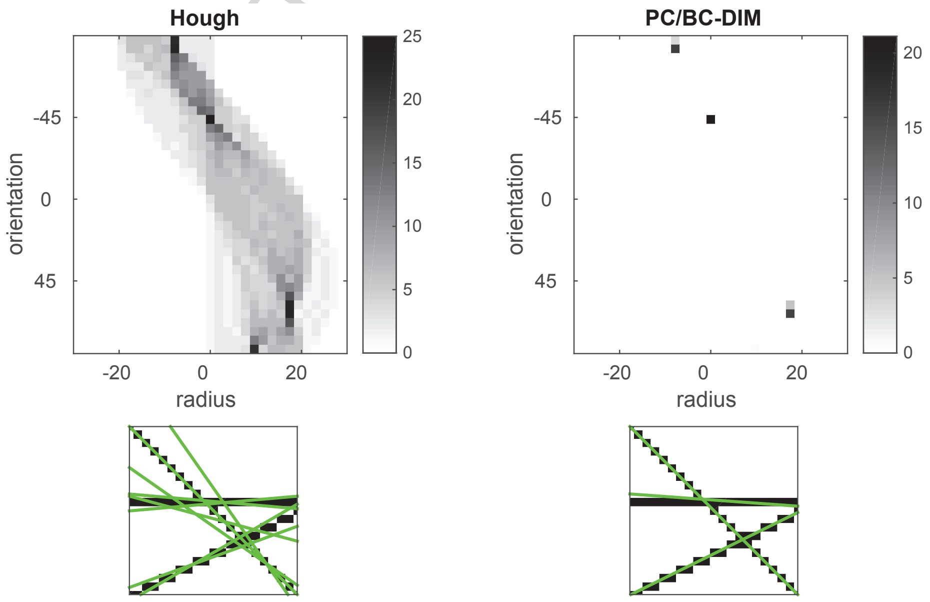

Figure 3: Results for the same image used in Fig. 1 when using a θ resolution of 5 and a ρ resolution of 2. The format is the same as, and described in the caption to, Fig. 1. Using a threshold of 12, standard Hough voting finds eight lines. PC/BC-DIM finds three lines with estimated parameters of (-85.9,-8), (-45,0), (63.8,18) that correspond to the lines in the image, although the parameters of the first line are poorly estimated.

These peaks all have amplitudes exceeding 20 and there are no other peaks with amplitudes greater than 0.001. It is therefore far easier to find a threshold that will identify the parameters of the lines in the image using the PC/BC-DIM method than using the standard Hough voting procedure.

The results shown in Fig. 1 were generated using an accumulator array with cells placed every 5^o degrees along the θ dimension and every 1 pixel along the ρ dimension. Figure 2 shows results for the same image when a different quantization of the accumulator array is employed. Specifically, the resolution in the θ dimension has been increased to 3.9^o degrees. In this case the estimate of the parameters of the -45^o line generated by the standard Hough voting mechanism is worse, due to -45^o not being represented by cells in the accumulator array. In contrast, the PC/BC-DIM algorithm represents the -45^o line using a small population of responses, and the exact parameters are successfully decoded from this distributed code. Another problem, that is not noticeable from Fig. 2, is that in the accumulator array generated by standard Hough voting the peaks that correspond to lines in the image are less distinguishable from spurious peaks. Specifically, while two cells corresponding to lines that are present in the image have values of 21, the third cell corresponding to the -45^o line has a value of 13. There are numerous other cells with high values, including 3 with a value of 9. In contrast, the amplitude of all three peaks found by PC/BC-DIM are in the range 19.9 to 21.3, and there are no other peaks with amplitudes greater than 0.001.

Figure 3 shows the effects of changing the quantization used for the ρ dimension of the accumulator array. In this case the resolution is 2 pixels. Using the same threshold as in the previous experiments (the highest threshold that can be used and still enable Hough voting to detect three lines in Fig. 2), eight lines are now detected using the standard Hough voting method. In contrast, the PC/BC-DIM method of voting succeeds in detecting the three lines in the image, although the parameters of one line are poorly estimated.

Figure 4 shows results for a more complex synthetic image. In contrast to the previous examples, this image is larger, contains more lines, and approximately 66% of the pixels are missing from each line. For this particular example, using the same threshold as previously, the PC/BC-DIM method performs better at detecting the lines than the standard Hough voting method. To determine if this was true in general, 250 images of this type were generated and each was analysed using the standard Hough voting method and the proposed one. Each image was 150 by 150 pixels and contained between four and eight lines. Roughly half the pixels in each line were removed. The parameters of the lines detected by the two methods were compared to the known parameters of the lines present in each image. To be considered a true-positive the value of θ given by the detector was required to be withing 4^o of the ground-truth value, and the value of ρ was required to be within 5 pixels of the ground-truth value. The precision recall curves for the two methods are shown in Fig. 4. It can be seen that the PC/BC-DIM method of voting significantly outperforms the standard Hough voting method when applied to this task. Specifically, the f-score was 0.99 for PC/BC-DIM, and 0.89 for Hough voting. Including additional pixels in each image to act as background noise makes the task of detecting the true lines much harder. However, as illustrated in Fig. 5 the performance of the PC/BC-DIM method of voting remains significantly superior to that of standard Hough voting. Specifically, with 1% noise the f-score was 0.96 for PC/BC-DIM, and 0.85 for Hough voting.

For the PC/BC-DIM method of voting, the parameters are determined by taking the weighted average of the parameters associated with all the cells forming an isolated peak in the accumulator array (see section 2.1). To determine if this population decoding method yielded more accurate parameter estimates than the standard HT, 250 randomly generated 150 by 150 pixel images were created, each of which contained exactly one straight line with approximately two-thirds of the edge pixels missing. Both the PC/BC-DIM method and the standard HT were applied to each of these images, and for both methods the parameters encoded by the peak with the most votes were found. The absolute difference between the estimated parameters and the true parameters were then calculated and the median value of this error across all 250 images was recorded. This experiment was repeated using accumulator arrays with various resolutions for the orientation parameter (θ) and the radius parameter (ρ). The results are shown in Fig. 6. It can be seen that, for all resolutions of the parameters, the population decoding method used

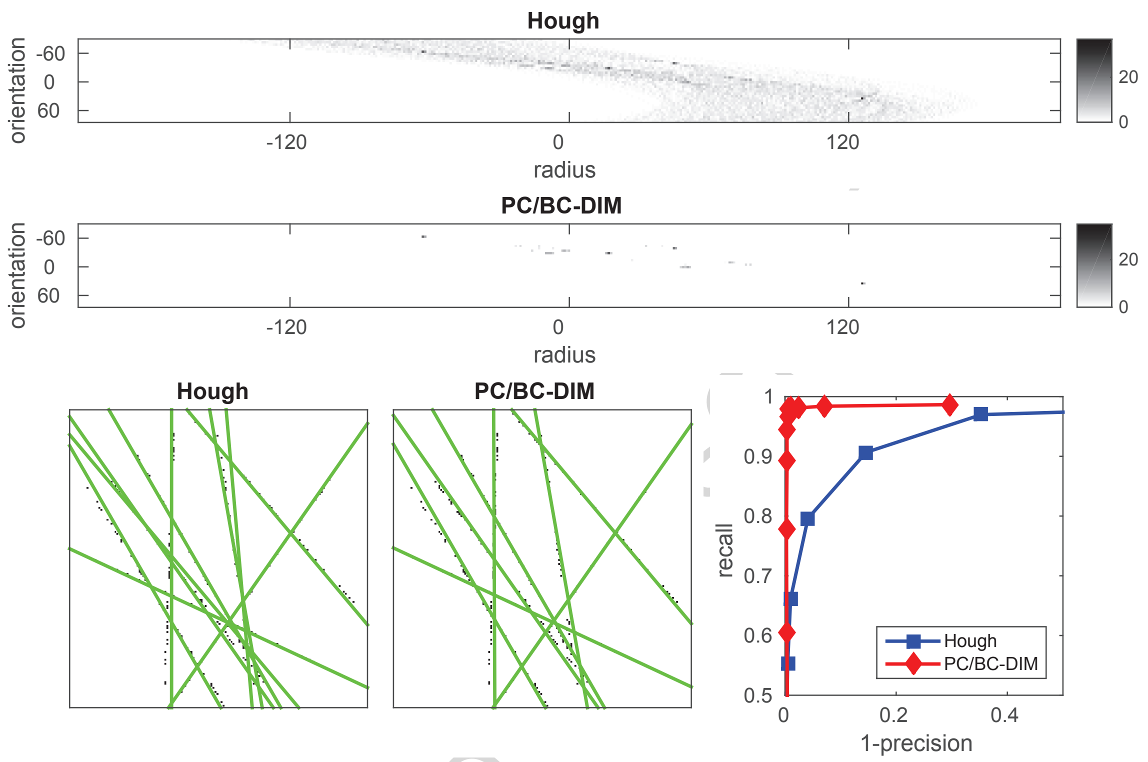

Figure 4: Results for a synthetic image containing eight lines with approximately 66% of the pixels missing. The top two rows show the accumulator arrays produced by standard Hough voting, and by the PC/BC-DIM method of voting when using a θ resolution of 5 and a ρ resolution of 1. The bottom row shows the original binary images (black pixels are edge pixels), and (in green) the lines detected by thresholding the corresponding accumulator array at a value of 12. Standard Hough voting detects 10 lines. PC/BC-DIM finds exactly the eight lines in the input image. The graph shows recall versus 1-precision for the standard Hough method and the proposed PC/BC-DIM method when applied to 250 randomly generated images each of which contains between four and eight lines with approximately two-thirds of the edge pixels missing.

by the proposed algorithm is more accurate at determining the parameters of the straight line.

3.2 Car detection

To assess the affects of using PC/BC-DIM to perform Hough voting in a natural image object detection task the method was applied to the UIUC cars dataset (Agarwal et al., 2004; Agarwal and Roth, 2002). An ISM-like detection method (see section 2.3) was trained on the training set which consists of 500 car images and 550 non-car images. For the single-scale task the test set contains 170 images containing 200 images of cars taken from the side. The multi-scale task has a test set of 108 images containing 139 cars at a variety of scales. For this latter task the ISM-like detection method was trained on the 500 car and 550 non-car training images resized to produced training images at six different scales.

To illustrate the behaviour of the proposed method, Fig. 7 shows the accumulator array produced by PC/BC-DIM in comparison to that produced by standard Hough voting for two images taken from the UIUC single-scale cars dataset. For the first image, the standard Hough voting scheme results in a peak corresponding to the location of the car, but also weaker peaks corresponding to locations where a car would be when the front wheel is identified as a rear wheel, and where a car would be when the rear wheel is identified as a front wheel. The PC/BC-DIM voting method removes these secondary peaks, as the votes that contribute to them have been explained away by the more likely explanation that the

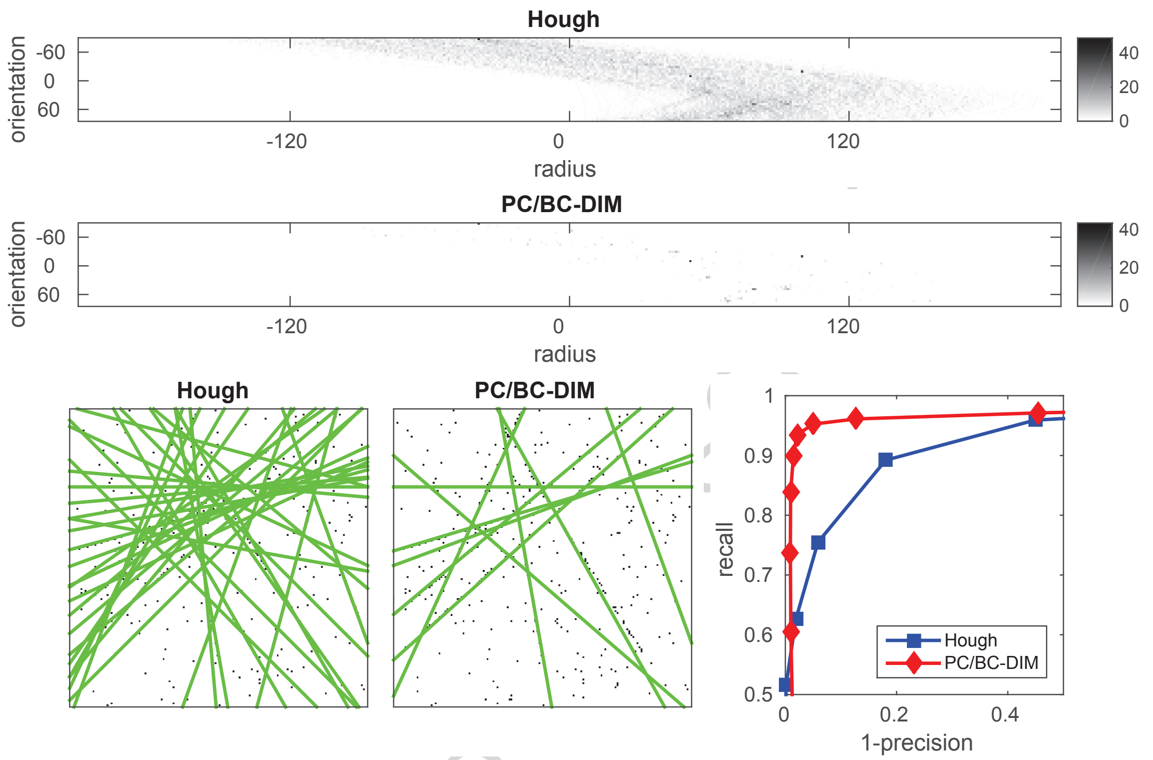

Figure 5: Results for a synthetic image containing eight lines with approximately 66% of the pixels missing and 1% of all pixels marked as edges to act as background noise. The format of the figure and the experimental parameters used are the same as, and described in the caption to, Fig. 4. Standard Hough voting produces seven true-positives, 27 false-positives, and one false-negative. PC/BC-DIM produces seven true-positives, three false-positives, and one false-negative. The graph shows recall versus 1-precision for the standard Hough method and the proposed PC/BC-DIM method when applied to 250 randomly generated images each of which contains between four and eight lines with approximately two-thirds of the edge pixels missing, and 1% of the background pixels added as noise.

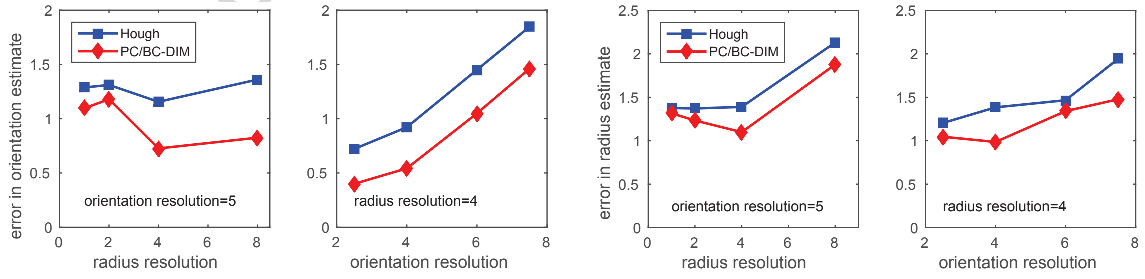

Figure 6: Median absolute error in parameter estimates for a range of resolutions of the accumulator array. At each resolution of the accumulator array, both the standard Hough method and the proposed PC/BC-DIM method were applied to 250 randomly generated images each of which contains exactly one line with approximately two-thirds of the edge pixels missing. The parameters of the line estimated by the largest peak in the accumulator array were compared to the true parameters of the line in the image. The graphs show the median absolute error accoss the 250 trials at each combination of parameter resolutions used. Errors in the estimate of the orientation parameter (θ) are shown in the two graphs on the left, and errors in the estimate of the radius parameter (ρ) are shown in the two graphs on the right.

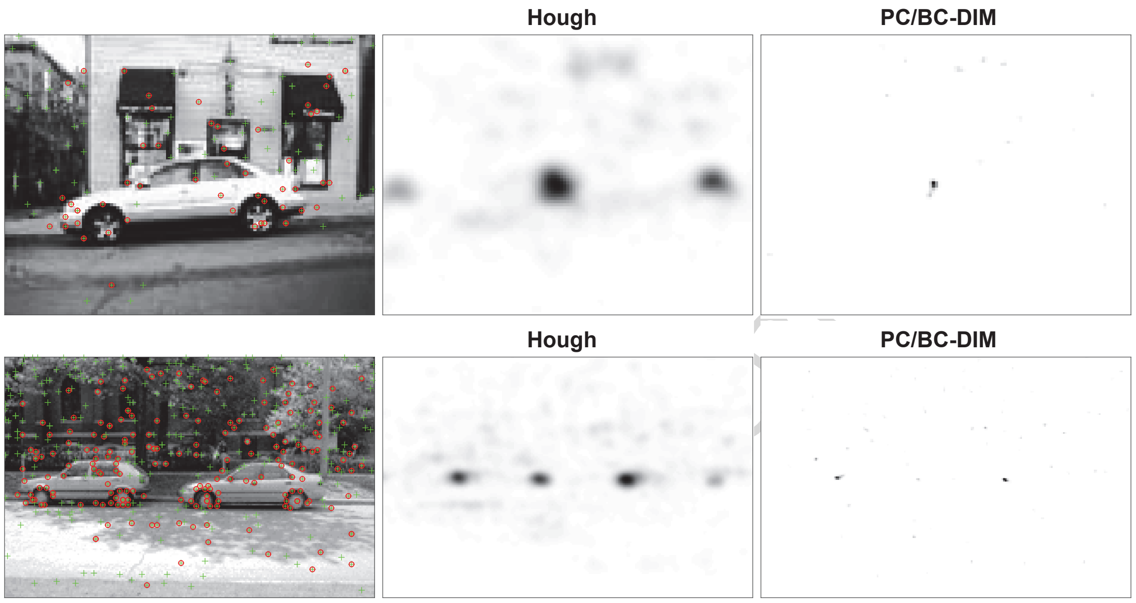

Figure 7: Results for two example images from the UIUC cars dataset. The first column shows the original test images, keypoints are indicated by the green crosses, and those keypoints at which image patches were matched to codebook entries corresponding to car parts are indicated by red circles. The second column shows the corresponding accumulator arrays produced by standard Hough voting, and the last column shows accumulator arrays generated by the PC/BC-DIM method.

Method	UIUC-single	UIUC-multi
Hough	17	33.1
PC/BC-DIM	1	3.6
ISM (Leibe et al., 2008)	9	-
ISM+MDL verification (Leibe et al., 2008)	2.5	5
Hough Forest (Gall and Lempitsky, 2009; Gall et al., 2011)	1.5	2.4
Discriminative HT (Okada, 2009)	1.5	-
IHRF (Lin et al., 2014)	0	1.3
PRISM (Lehmann et al., 2011)	-	2.2
ESS (Lampert et al., 2008)	1.5	1.4
sliding window HMAX+verification (Mutch and Lowe, 2006)	0.06	9.4
chains model (Karlinsky et al., 2010)	0.5	-

Table 1: Percentage EER for various methods on the UIUC single-scale and multi-scale cars dataset.

car is in the centre of the image. For the second image, the standard method of voting results in four peaks. The left-most corresponds to the location voted for by the front and rear wheels of the left-most car. The second peak corresponds to a location voted for by the rear wheels of the left-hand car and the front wheels of the right-hand car. The third peak corresponds to the location voted for by the front and rear wheels of the right-most car, and the right-most peak is at the second location voted for by the rear wheels of the right-hand car. The PC/BC-DIM voting method explains-away the false peaks, leaving two clear maxima corresponding to the locations of the two cars.

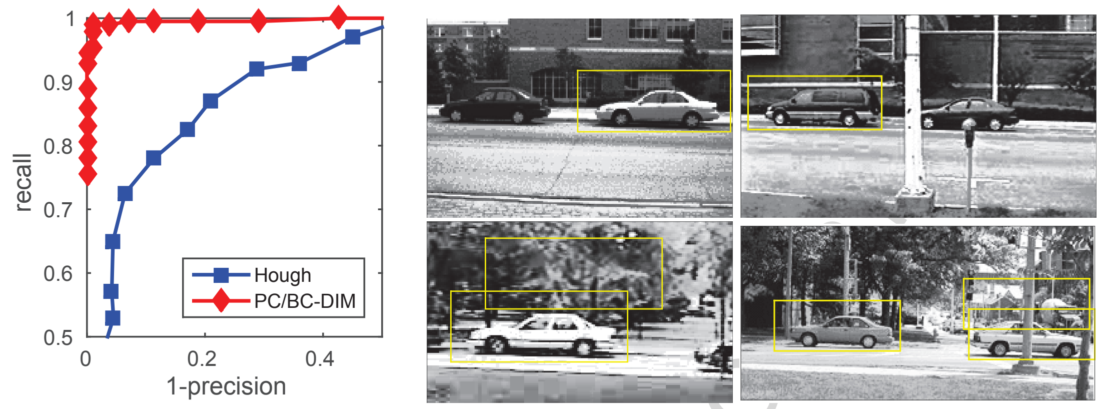

Figure 8: Results of applying the proposed method to the single-scale UIUC cars dataset. The graph shows recall versus 1-precision for the standard Hough voting method and the proposed PC/BC-DIM method. The images show all the failure cases at equal error rate for the PC/BC-DIM method. Bounding boxes for cars detected by the proposed algorithm are shown in yellow. The top row shows the two cases in which the proposed method fails to locate a car. The bottom row shows the two cases where the proposed method makes a false detection.

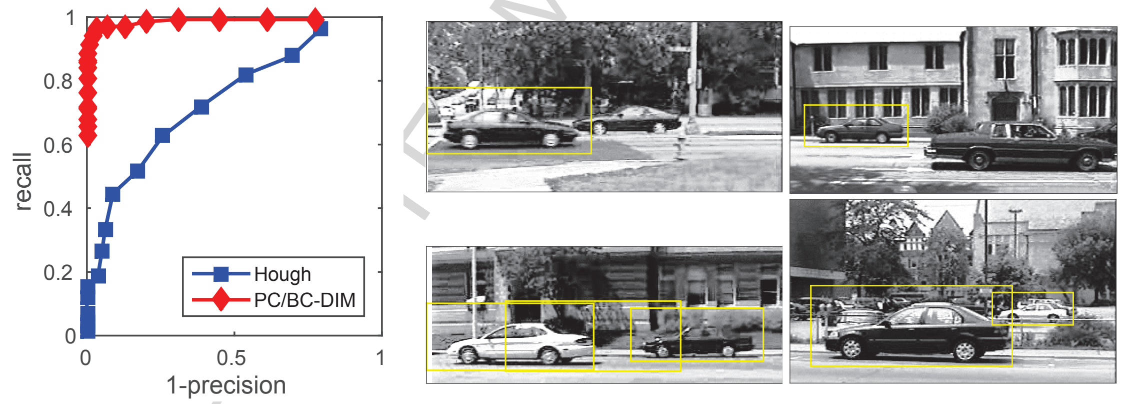

Figure 9: Results of applying the proposed method to the multi-scale UIUC cars dataset. The graph shows recall versus 1-precision for the standard Hough voting method and the proposed PC/BC-DIM method. The images show four, out of the 10, failure cases at equal error rate for the PC/BC-DIM method. Bounding boxes for cars detected by the proposed algorithm are shown in yellow. The top row shows two cases in which the proposed method fails to locate a car. The bottom row shows two cases where the proposed method makes a false detection. Note that the last image has been flagged as containing a false-positive as the lighter, more distant, car is not included as a true-positive in the annotation of the test data (the same situation occurs for another test image, not shown).

To quantitatively assess the performance of the proposed algorithm the criteria for counting true-positives, false-positives, and false-negatives defined by Agarwal et al. (2004) was used (indeed, the java code supplied with UIUC cars dataset was used to perform this evaluation). The precision recall curves obtained on the UIUC single-scale cars dataset by the PC/BC-DIM method of voting and the standard Hough method are shown in Fig. 8. It can be seen that the PC/BC-DIM method is significantly more accurate than standard Hough voting. Specifically, the f-score was 0.99 for PC/BC-DIM and 0.83 for Hough voting. The EER was 1% for PC/BC-DIM and 17% for Hough voting. The results obtained on the UIUC multi-scale cars dataset are shown in Fig. 9. In this case also, the proposed method of voting is significantly more accurate at detecting cars than standard Hough voting. Specifically, the f-score was 0.96 for PC/BC-DIM and 0.68 for Hough voting. The EER was 3.6% for PC/BC-DIM and 33.1% for Hough voting. These results are compared to other published methods in Table 1. It can be seen that the proposed method is competitive with the state-of-the art, and particularly, that it outperforms ISM.

The ISM algorithm replaces the standard Hough voting procedure with a method that casts votes in a continuous Hough space, then uses the mean-shift clustering algorithm to find peaks, and subsequently uses an MDL verification method to reject false peaks. Here, the standard Hough voting procedure has been replaced with the PC/BC-DIM algorithm, and this gives rise to larger improvement in performance over the standard Hough voting method than is seen for ISM. However, it is not possible to attribute this improved performance solely to the new voting procedure as the implementation of an ISM-like algorithm given in section 2.3 has a number of other differences to the version of the ISM originally proposed by Leibe et al. (2004, 2008). Specifically, the ISM-like algorithm used here is trained using the UIUC training set rather than a bespoke dataset. The results for ISM reported in Table 1 were produced using different keypoint detectors and image descriptors for the single-scale and multi-scale tasks. Specifically, for the single-scale dataset ISM used the Harris keypoint detector and image features that were image patches. For the multi-scale dataset ISM used Hessian-Laplace keypoints and Local Shape Context features (Belongie et al., 2002). In contrast, the ISM-like algorithm used here employed the same detector and descriptor for both tasks: a combination of both the Harris and SIFT detectors and grayscale image patches. To generate the codebook, ISM uses average-link agglomerative clustering and retains all clusters which have at least two members, while the ISM-like algorithm used here employs complete-link agglomerative clustering, and retains only those clusters that have at least 13 members. In the original ISM algorithm, during detection, the descriptor around each keypoint can be matched to all codebook entries that have a similarity exceeding a threshold. In the ISM-like algorithm implemented for this work, at most one codebook entry is matched to each keypoint, and the method of determining if a codebook entry matches or not uses a second codebook trained on the background images.

4 Discussion

In the Hough Transform, each image element will generate multiple votes. For example, a pixel that has been identified as an edge will vote for multiple orientations of straight line that can pass through that point, or a patch of image that has been identified as part of a car will vote for multiple places were the car centre might be located. However, typically only one (or none) of these votes will be correct. There will typically be at most one straight line passing through the edge pixel, and there will be at most one car that the image patch forms part of. If the correct vote can be identified then this can be used to "explain away" the other, spurious, votes removing them from the accumulator array. One consequence of explaining away is, therefore, that it is far easier to identify the peaks in the accumulator which correspond to shapes which are present in the image (*i.e.*, it helps to solve the problem of "spurious peaks"). A second consequence of explaining away is that the accumulator array becomes much sparser, allowing the parameters of the shape to be calculated as the weighted average of the parameters associated with all the cells forming an isolated peak in the accumulator array. This can result in a more accurate estimation of the parameters and for parameter values to be interpolated between the those represented by individual accumulator array cells (*i.e.*, it helps to solve the problem of "quantization effects").

This article proposes a new method for performing explaining away in the HT. This novel method utilises a competitive neural network algorithm called PC/BC-DIM. This algorithm performs explaining away through an iterative process. It has been shown that the proposed method results in a significant improvement in performance over standard Hough voting in two applications: straight line detection, and car detection in natural images.

The current work has been concerned with improving the Hough voting process. Other recent work on the HT has been concerned with improving the identification of voting elements (*e.g.*, Gall and Lempitsky, 2009; Gall et al., 2011; Lin et al., 2014; Maji and Malik, 2009; Okada, 2009; Razavi et al., 2011). This previous work is entirely complementary to the current work, and it should therefore be possible to combine the proposed voting process with these improved methods of voting element identification to further boost the performance of the HT. For example, here an unsupervised clustering method was used to define a codebook, codebook entries were grayscale image patches, and these were matched to a sparse set of image locations. However, recent work on improving the identification of voting elements suggests that performance is improved by using a supervised learning method to define a descriminative codebook, using image descriptors other than grayscale intensity, and using dense sampling of the images.

References

Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–90.

Agarwal, S. and Roth, D. (2002). Learning a sparse representation for object detection. In *Proceedings of the European Conference on Computer Vision*, volume IV, pages 113–30.

Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–22.

Barinova, O., Lempitsky, V., and Kohli, P. (2012). On detection of multiple object instances using Hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–84.

Beinglass, A. and Wolfson, H. J. (1991). Articulated object recognition, or: how to generalize the generalized Hough transform. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 461–6.

Belongie, S., Malik, J., and Puchiza, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–22.

Davies, E. R. (1988). Application of the generalised Hough transform to corner detection. *IEE Proceedings E (Computers and Digital Techniques)*, 135(1):49–54.

Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–5.

Furukawa, Y. and Shinagawa, Y. (2003). Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space. *Computer Vision and Image Understanding*, 92(1):1–25.

Gall, J. and Lempitsky, V. (2009). Class-specific Hough forests for object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–202.

Georgopoulos, A. P., Schwartz, A. B., and Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, 233:1416–9.

Gerig, G. (1987). Linking image-space and accumulator-space: A new approach for object-recognition. In *Proceedings of the International Conference on Computer Vision*, pages 112–7.

Harpur, G. and Prager, R. (1996). Development of low entropy coding in a recurrent network. *Network: Computation in Neural Systems*, 7(2):277–84.

Harpur, G. F. and Prager, R. W. (1994). A fast method for activating competitive self-organising neural networks. In *Proceedings of the International Symposium on Artificial Neural Networks*, pages 412–8.

Hart, P. E. (2009). How the Hough transform was invented [DSP history]. *IEEE Signal Processing Magazine*, 26(6):18–22.

Hough, P. V. C. (1962). Method and means for recognizing complex patterns. U.S. Patent 3 069 654.

Illingworth, J. and Kittler, J. (1987). The adaptive Hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):690–8.

Ji, J., Chen, G., and Sun, L. (2011). A novel Hough transform method for line detection by enhancing accumulator array. *Pattern Recognition Letters*, 32(11):1503 – 1510.

Karlinsky, L., Dinerstein, M., Daniel, H., and Ullman, S. (2010). The chains model for detecting parts by their context. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Kersten, D., Mamassian, P., and Yuille, A. (2004). Object perception as Bayesian inference. *Annual Review of Psychology*, 55(1):271–304.

Lampert, C., Blaschko, M., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Lehmann, A., Leibe, B., and Van Gool, L. (2011). Fast prism: Branch and bound Hough transform for object class detection. *International Journal of Computer Vision*, 94(2):175–197.

Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *European Conference on Computer Vision Workshop on Statistical Learning in Computer Vision*.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–89.

Lin, Y., Lu, N., Lou, X., Zou, F., Yao, Y., and Du, Z. (2014). Invariant Hough random ferns for object detection and tracking. *Mathematical Problems in Engineering*, 2014(513283):20.

Lochmann, T. and Deneve, S. (2011). Neural processing as causal inference. *Current Opinion in Neurobiology*, 21(5):774–81.

Lochmann, T., Ernst, U. A., and Denève, S. (2012). Perceptual inference predicts contextual modulations of sensory responses. *The Journal of Neuroscience*, 32(12):4179–95.

Maji, S. and Malik, J. (2009). Object detection using a max-margin Hough transform. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Mutch, J. and Lowe, D. G. (2006). Multiclass object recognition with sparse, localized features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11–18, New York, NY.

Okada, R. (2009). Discriminative generalized Hough transform for object dectection. In *Proceedings of the International Conference on Computer Vision*, pages 2000–2005.

Rao, R. P. N. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87.

Razavi, N., Gall, J., and van Gool, L. (2011). Scalable multi-class object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1505–12.

Samal, A. and Edwards, J. (1997). Generalized Hough transform for natural shapes. *Pattern Recognition Letters*, 18:473–80.

Spratling, M. W. (1999). Pre-synaptic lateral inhibition provides a better architecture for self-organising neural networks. *Network: Computation in Neural Systems*, 10(4):285–301.

Spratling, M. W. (2008). Predictive coding as a model of biased competition in visual selective attention. *Vision Research*, 48(12):1391–408.

Spratling, M. W. (2010). Predictive coding as a model of response properties in cortical area V1. *The Journal of Neuroscience*, 30(9):3531–43.

Spratling, M. W. (2012a). Predictive coding as a model of the V1 saliency map hypothesis. *Neural Networks*, 26:7–28.

Spratling, M. W. (2012b). Unsupervised learning of generative and discriminative weights encoding elementary image components in a predictive coding model of cortical function. *Neural Computation*, 24(1):60–103.

Spratling, M. W. (2013). Image segmentation using a sparse coding model of cortical area V1. *IEEE Transactions on Image Processing*, 22(4):1631–43.

Spratling, M. W. (2014a). Classification using sparse representations: a biologically plausible approach. *Biological Cybernetics*, 108(1):61–73.

Spratling, M. W. (2014b). A single functional model of drivers and modulators in cortex. *Journal of Computational Neuroscience*, 36(1):97–118.

Spratling, M. W., De Meyer, K., and Kompass, R. (2009). Unsupervised learning of overlapping image components using divisive input modulation. *Computational Intelligence and Neuroscience*, 2009(381457):1–19.

Spratling, M. W. and Johnson, M. H. (2002). Pre-integration lateral inhibition enhances unsupervised learning. *Neural Computation*, 14(9):2157–79.

Spratling, M. W. and Johnson, M. H. (2003). Exploring the functional significance of dendritic inhibition in cortical pyramidal cells. *Neurocomputing*, 52-54:389–95.

Woodford, O., Pham, M.-T., Maki, A., Perbet, F., and Stenger, B. (2014). Demisting the Hough transform for 3D shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–41.

Xu, Z., Shin, B.-S., and Klette, R. (2015). Accurate and robust line segment extraction using minimum entropy with Hough transform. *Image Processing, IEEE Transactions on*, 24(3):813–822.

Yao, A., Gall, J., and Gool, L. V. (2010). A Hough transform-based voting framework for action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2061–8.