

Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy

Marcos Nieto · Carlos Cuevas · Luis Salgado · Narciso García

Received: 4 August 2009 / Accepted: 26 March 2011 / Published online: 9 April 2011
© Springer-Verlag London Limited 2011

Abstract A new line segment detection approach is introduced in this paper for its application in real-time computer vision systems. It has been designed to work unsupervised without any prior knowledge of the imaged scene; hence, it does not require tuning of input parameters. Although many works have been presented on this topic, as far as we know, none of them achieves a trade-off between accuracy and speed as our strategy does. The reduction of the computational cost compared to other fast methods is based on a very efficient sampling strategy that sequentially proposes points on the image that likely belong to line segments. Then, a fast line growing algorithm is applied based on the Bresenham algorithm, which is combined with a modified version of the mean shift algorithm to provide accurate line segments while being robust against noise. The performance of this strategy is tested for a wide variety of images, comparing its results with popular state-of-the-art line segment detection methods. The results show that our proposal outperforms these works considering simultaneously accuracy in the results and processing speed.

Keywords Line segment · Eigenvalues · Real time · Slice sampling · Mean shift · Bresenham algorithm

1 Introduction

Straight lines or line segments within an imaged scene can be of great help to infer its geometric properties and also the projection process that converts 3D world elements into a 2D image. Man-made environments typically contain multiple line segments oriented towards a number of common directions (belonging to flat surfaces such as the ground, doors, walls, or buildings). For that reason, line segments can be used as low level features for conventional computer vision problems, such as the detection of vanishing points [2, 20], autocalibration [27], plane rectification techniques [21, 23], 3D reconstruction [22], object tracking [29] or active contours tracking [30].

This paper models the problem of finding line segments in real images in a probabilistic way. An image is considered as a set of observations $I = \{(x_k, y_k)\}_{k=1\dots N}$ coming from a sub-jacent probability density function $p(x, y)$ that represents the probability of a pixel to belong to a line segment. One of the main contributions of the paper is on the generation of an estimate $\hat{p}(x, y)$ of this likelihood distribution from the analysis of the eigenvalues of the tensor matrix associated to the image pixels. Besides, the real-time performance of the proposed strategy is achieved by the use of a sequential sampling technique that selects pixels in the image that likely belong to line segments according to $\hat{p}(x, y)$. Line segments are obtained through the Bresenham growing algorithm enhanced with the use of weighted mean shift (wMS) procedures that provides accurate fits and robustness against noise.

The target of our method is the obtention of line segments in any kind of real images. Therefore, Sect. 7

M. Nieto (✉)
Vicotech-ik4, Mikeletegi Pasealekua 57,
20009 Donostia-San Sebastián, Spain
e-mail: mnieto@vicotech.org

C. Cuevas · L. Salgado · N. García
Grupo de Tratamiento de Imágenes,
Universidad Politécnica de Madrid, 28040 Madrid, Spain
e-mail: ccr@gti.ssr.upm.es

L. Salgado
e-mail: L.Salgado@gti.ssr.upm.es

N. García
e-mail: narciso@gti.ssr.upm.es

evaluates the obtained line segments for a wide variety of real images. Nevertheless, since one of the major feats of our approach is its low computational cost, its performance is better exploited for vision applications working on sequences of images. For instance, camera autocalibration strategies based on the computation of vanishing points in moving camera sequences of structured environments [20] require real-time operation for accurate and meaningful line segment detections.

2 Related work

Most approaches in the related literature use edges extracted at pixel level from the images as basic information. The Sobel edge detector [16] is typically used as an approximation to the first order spatial derivatives $I_x(x, y)$ and $I_y(x, y)$. For each pixel, k at location (x_k, y_k) the module and orientation of the gradient can also be estimated as described in [9].

Once obtained, the information of edges at pixel level can be used in different ways to search for line segments. Two main groups of techniques can be identified: those based on accumulation in parametric spaces, such as methods based on the Hough transform (HT) [1, 3, 9, 11, 17, 19, 31]; and pixel clustering strategies in the image plane [6, 13, 14, 18].

2.1 Hough transform

The former group basically performs a parametric transformation to the set of edge pixels, from which lines are extracted searching for local maxima in the transform domain. Note that the standard HT (SHT) detects lines, but not line segments, which require addressing additional considerations for their detection, as done in [18, 35]. Other works face the problem of finding lines, as done by [19, 31, 34] or recently Aggarwal [1] with the Regularized HT.

The major drawback of the works based on the HT is the excessive algorithm complexity [9, 31], which avoids its use on online applications.

Random sampling applied to the HT (RHT) has been originally proposed by Xu et al. [34], and further improved by Kiryati et al. [19] to render efficient line segment detectors. A pair of edge points is selected at random to accumulate a single vote on the transformed domain. These approaches are the so-called “many-to-one” voting schemes, in contrast to the “one-to-many” corresponding to the standard HT, which dramatically enhances the speediness of the HT. Several variants of the RHT have been proposed by Kälviäinen et al. [17], which improve the sampling distribution by means of a two-step random process. Windows of fixed or also random size are

randomly selected in the image, and the RHT is then applied to search for maxima. Walsh and Raftery [32] proposed an importance sampling procedure to improve the random sampling used within the RHT.

The resulting line segments can be further grouped into longer ones as done by Bandera et al. [3], which uses mean shift procedures to cluster line segments in the transform domain and obtain a single representative for each cluster.

Nevertheless, the main problem of these approaches is that they achieve fast results at the cost of reducing their accuracy, giving a large number of false negatives (miss-detections due to the termination criteria), especially in very fast detectors as the PPHT (Progressive Probabilistic HT) by Matas et al. [11]. Besides, they require a large set of application-dependent threshold values to be tuned to obtain adequate results. As final remark, the need of such a number of user-defined thresholds in this type of approaches reduces its applicability to unsupervised systems as well as making them more prone to errors.

2.2 Pixel clustering

This group first cluster sets of connected pixels in a coarse manner according to some common property (e.g. their orientation) and afterwards compute an estimate for each set. The work by Burns et al. [6] has been used as reference by further authors [13, 18]. Besides, the use of connected component analysis (CCA) was first introduced in this context by Nevatia and Babu [26] and followed by other authors [18, 20]. Analogously, Yuen et al. [35] propose a connected version of the HT, which selects at random an edge pixel and then apply an one-dimensional accumulation on the angle parameter.

In this field, some of the works exploit the information contained in the eigenvalues and vectors of the image tensor matrix [33], as done by many others [14, 18, 20]. Typically, the straightness of the group of pixels is measured as a function on the eigenvalues (λ_1, λ_2) . However, it is not straightforward to classify a pixel, given its eigenvalues, into these three categories. As mentioned by Rosten et al. [28], many authors have created unbounded functions that evaluate the “cornerness” of pixels by the computation of λ_2/λ_1 , λ_2 , or approximations to the relationship between eigenvalues that skip their computation using directly the elements of the tensor matrix, such as the well-known Harris corner function.

3 Approach overview

The proposed strategy gathers properties of the two reviewed groups of line detection methods, although it cannot be classified as belonging to any of them. On the

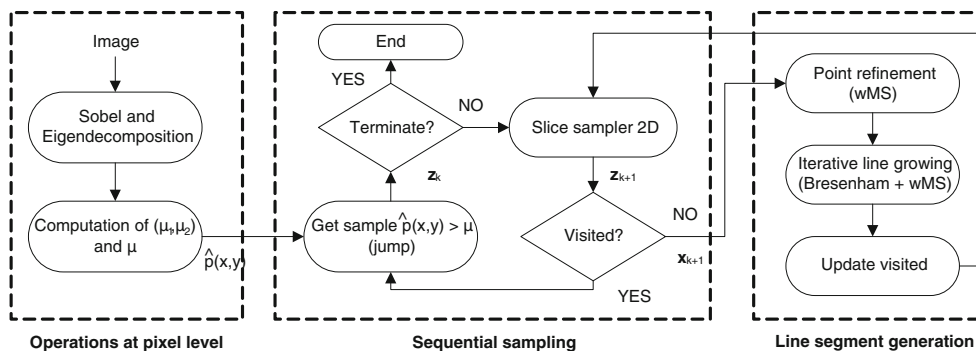


Fig. 1 Flow chart of the proposed strategy. The image is processed in order to obtain the likelihood distribution $\hat{p}(x, y)$ defined by the parameters (μ_1, μ_2) . The mean value of $\hat{p}(x, y)$, μ , is used to generate the first sample of the algorithm and start running the slice sampler,

which sequentially generates samples, z_k . The last stage is the line segment generation, in which the samples are refined and used as starting point for the iterative line growing algorithm that finally generates the line segment

one hand, it uses a sequential sampling strategy, which is an improved version of the random sampling approaches used by Hough-based methods. On the other hand, the proposed method uses the information of the second moment matrix at pixel level as also done by some clustering-based methods, although in a novel way, since it computes a new likelihood function $\hat{p}(x, y)$.

The flow chart of the proposed method (which will be denoted as SSWMS, slice sampling weighted mean shift), is depicted in Fig. 1. As shown, it is composed of three main stages: the computation of the likelihood function, the sequential sampling, and the line segment generation.

The first step estimates the likelihood distribution over the image I such that $\hat{p}(x, y)$ is the likelihood value of a pixel (x, y) to belong to a line segment. This distribution is parameterized by (μ_1, μ_2) , which are statistics automatically obtained from the image. The distribution is defined using the image tensor matrix, and their corresponding eigenvalues. This process is guided by the idea that pixels that belong to line segments must satisfy two criteria: (1) they have significant gradient magnitude; and (2) there is only one dominant direction in their neighborhoods. These concepts and the associated methods are described in detail in further sections.

The sequential sampling, based on the slice sampling algorithm [4, 25], sequentially selects pixels in the image, denoted as $z_k = (x_k, y_k)$, that are good candidates to belong to line segments, according to the computed likelihood $\hat{p}(z_k)$. The design of this stage ensures that no repeated samples are selected, so that $z_k \neq z_c \forall c < k$. If the slice sampler proposes a sample z_k that has been already visited by the algorithm, the next sample is selected randomly satisfying $\hat{p}(x, y) > \mu$, where μ is the mean value of $\hat{p}(x, y)$ computed for all the pixels of the image.

Given a candidate pixel proposed by the sampling technique, the process arrives to the line segment generation stage. It starts with the point refinement module, which

uses mean shift (MS) procedures on a multidimensional space composed of the position of the pixels and their dominant local orientation, denoted as $x_k = (x_k, y_k, \theta_k)$ ¹. The MS searches for a local maxima on this space, i.e., it looks for the pixel in the neighborhood of the candidate pixel that most likely belongs to a line segment.

From this local maxima, an efficient line growing strategy, based on the Bresenham algorithm is applied to connect pixels until the end points of the line segment are reached. This scheme is applied iteratively to enhance the accuracy of the generated line segments. All the pixels swept by the generated line segment are marked as visited, and thus not available to be selected anymore during the sequential sampling stage.

The overall result of the strategy is that the sequential selection of candidates with the slice sampler is much more efficient than processing the whole image in search for line segments (as done by most works based on pixel clustering techniques), and even more than randomly selecting points on the image without any guiding criteria (as the works about random sampling on the Hough domain). On the other hand, the accuracy on the line segment-fitting process is provided by the MS procedure that is used to refine the candidates and generate good starting and ending points for the growing algorithm.

4 Sequential sampling strategy

In this section we describe the sequential sampling stage, which is based on a general sampling algorithm: the slice sampler. Also, the computation of the likelihood function for line segments and the initialization and termination processes are presented.

¹ For the sake of clarity in the notation, italic characters correspond to scalar values while bold characters represent arrays.

4.1 Slice sampling

The slice sampler [25] is a general sampling strategy born in the field of inference methods based on numerical sampling (typically known as Markov Chain Monte Carlo techniques) [4]. It allows to sequentially obtain samples, $\{\mathbf{z}_k\}_{k=1}^N$ from a arbitrary target pdf, $p(\mathbf{z})$. The only requirement to apply this algorithm is that the value $p(\mathbf{z})$ shall be evaluated for any given value of \mathbf{z} .

As described in [4], the slice sampling improves the results, in terms of efficiency, of typical sampling approaches based on the Metropolis–Hastings (MH) algorithm [12]. This algorithm (MH) has an important drawback that makes it inefficient for the proposed line segment detector as it is sensible to the step size, given by the proposal distribution. If it is chosen too small, the process behaves as a random walk, which makes the algorithm converge very slowly and, on the contrary, if it is too large, the rejection rate may be very high, hence not achieving accurate results. The advantage of the slice sampler is due to its ability to automatically adapt its step size according to the characteristics of the pdf.

For a better understanding of the slice sampler, let us first consider the univariate case: $p(z)$. Slice sampling works by augmenting z with an auxiliary random variable u and then sample from the joint (z, u) space [25]. Given the previous sample z_{k-1} , u is uniformly drawn in the range $[0, p(z_{k-1})]$. Fixed u , the sample z_k is obtained from the “slice” through the distribution defined by $\{z: p(z) > u\}$. This criterion is illustrated in Fig. 2a. Nevertheless, it is difficult to find the limits of the slice and thus to draw a sample from it. For that reason an approximation is done by means of creating a quantized local slice, delimited by z_0 and z_1 as shown in Fig. 2b. To obtain these limits, the value $p(z)$ is evaluated at left and right of z_{k-1} using fixed length steps (the quantification step) until $p(z) < u$. The next sample, z_k , is obtained by uniformly sampling on this range (iteratively until $p(z_k) > u$).

For the line segment detection, the sampling has to be carried out on a two-dimensional space. We propose to obtain samples by sequentially applying one-dimensional slice sampling at each dimension, x and y . Figure 3 illustrates this procedure, depicting the contour plot of a zoom on a two-dimensional, $p(\mathbf{z})$, function. For a given sample $\mathbf{z}_{k-1} = (x_{k-1}, y_{k-1})$, depicted as a black square, the one-dimensional slice criterion is applied first on x : y_{k-1} is fixed and a new x_k is delivered. This value is then fixed (represented as a circle) and the one-dimensional procedure is repeated for y . The result is the new two-dimensional sample $\mathbf{z}_k = (x_k, y_k)$. Hence, the 2D slice sampler used is a combination of two 1D-slice steps, which results in a fast and efficient 2D sampling. As a remark, note that if the order of the one-dimensional

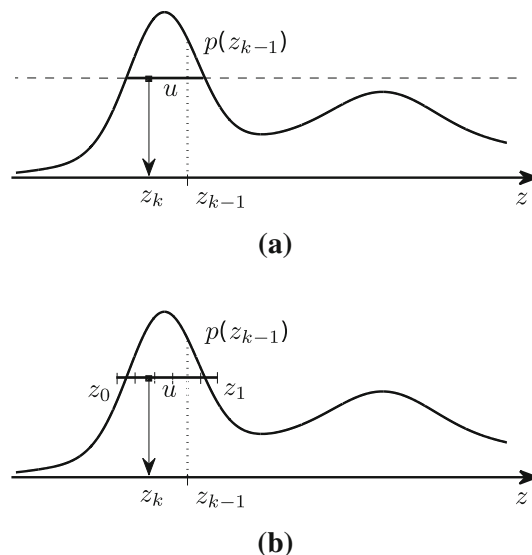


Fig. 2 Univariate slice sampling: **a** the uniform u value determines the slice through $p(z)$; and **b** the practical implementation uses fixed length steps to determine the range in which z is uniformly sampled

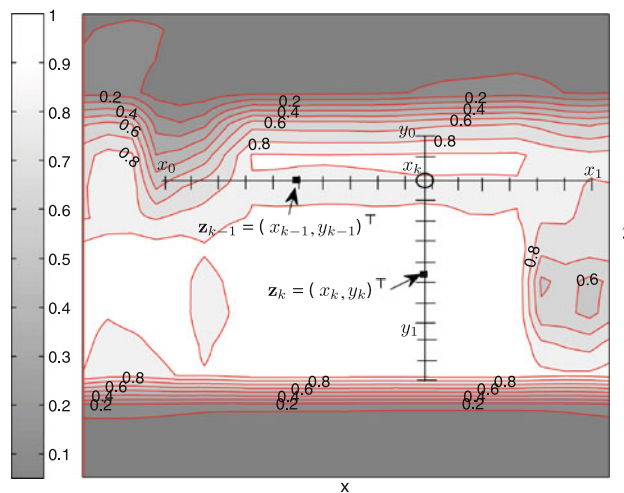


Fig. 3 The level set plot allows to observe that \mathbf{z}_{k-1} is located in a region with levels between 0.6 and 0.8. The slice procedure on x produces the region limited by x_0 and x_1 . The selected sample x_k is fixed and the region between y_0 and y_1 is obtained for the y dimension. The final sample is $\mathbf{z}_k = (x_k, y_k)$

samplers is reversed, a different final sample \mathbf{z}_k could be generated, though this is irrelevant for the line segment generation, as the mean shift procedures refine these samples considering a similar neighborhood.

4.2 Line segment likelihood

The slice sampling algorithm requires the construction of a target pdf, which is, in this case, the likelihood of the image pixels to belong to line segments.

Different methods have been addressed in the literature to characterize image pixels to belong to corners, homogeneous regions or line segments. Most of them use the covariance matrix or image tensor matrix for each pixel and their associated eigenvalues, (λ_1, λ_2) with $\lambda_1 > \lambda_2$, and eigenvectors, $(\mathbf{e}_1, \mathbf{e}_2)$ (details can be found in works as [33]).

The eigenvalues illustrate the dispersion of the gradient along their associated eigenvectors [28] as can be seen in Fig. 4. For that reason, a point that belongs to a line segment is described by a significant value of λ_1 , while λ_2 is close to zero. Analogously, a corner is given by a pair of eigenvalues with similar magnitude both being far from zero, whereas pixels inside homogeneous regions have both eigenvalues close to zero. These cases are depicted in Fig. 4, where the obtained eigenvalues are represented as the axis of the ellipse fitting the gradient distribution of the neighborhood of the considered pixel.

In this work, we propose to handle the eigenvalues information by means of a novel function composition, based on the eigenvalues of the pixels of the whole image to generate a function that maps back the likelihood value of a pixel given its particular eigenvalues. The function satisfies two criteria: on the one hand, it returns high values only for pairs of eigenvalues for which one of them is much higher than the other; and, on the other hand, it decreases rapidly when the eigenvalues are both high or low. The function is defined as:

$$p = g \circ f : I \rightarrow (\lambda_1, \lambda_2) \rightarrow \mathbb{R} \quad (1)$$

$$(x, y) \mapsto \hat{p}(x, y) = g(f(x, y))$$

where $f(x, y)$ is the function that gives, for each pixel (x, y) , the pair of corresponding eigenvalues (λ_1, λ_2) , and $g(\lambda_1, \lambda_2)$ is the function that determines the likelihood of a pair of eigenvalues to correspond to a line segment, defined as:

$$g(\lambda_1, \lambda_2) = (1 - \exp(-\lambda_1/\mu_1)) \exp(-\lambda_2/\mu_2) \quad (2)$$

where the parameters (μ_1, μ_2) are, respectively, the average values of the eigenvalues computed over the whole image, and have been obtained in the first stage of the proposed

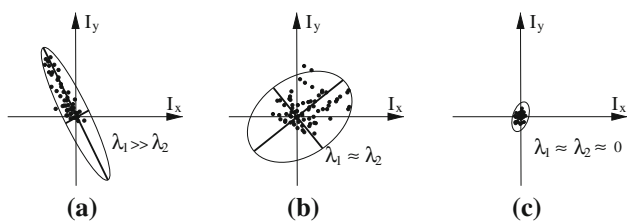


Fig. 4 Eigenvalues illustrating gradient structures. *Black dots* are the (I_x, I_y) values of the neighbor pixels of an example pixel that corresponds to: **a** a line segment, **b** a corner or an heterogeneous gradient region, and **c** homogeneous region

strategy. The function $g(\lambda_1, \lambda_2)$ is the product of two independent functions on each eigenvalue. The first term, on λ_1 , penalizes the response for those pixels having their largest eigenvalue too small, i.e., those that likely belong to an homogeneous region. The second term, on λ_2 , enhances the response for those pixels with a very low smallest eigenvalue, thus penalizing those corresponding to corners, which are described by high values of λ_1/μ_1 and λ_2/μ_2 . The result is a good representation of the likelihood of a pixel to belong to a line segment. This function shows high response values for pixels with a significant largest eigenvalue, and a very low smallest one. For one example image, this function is illustrated in Fig. 5a, while the associated scatter plot of (λ_1, λ_2) is shown in Fig. 5b.

As an example, the $\hat{p}(x, y)$ values of all the pixels in an image have been computed. Fig. 6 shows, in (a) the original image, which contains large homogeneous regions, significant line segments and corners; (b) shows an scaled representation of the line segment likelihood, where pixels with higher intensity have higher probability to belong to line segments.

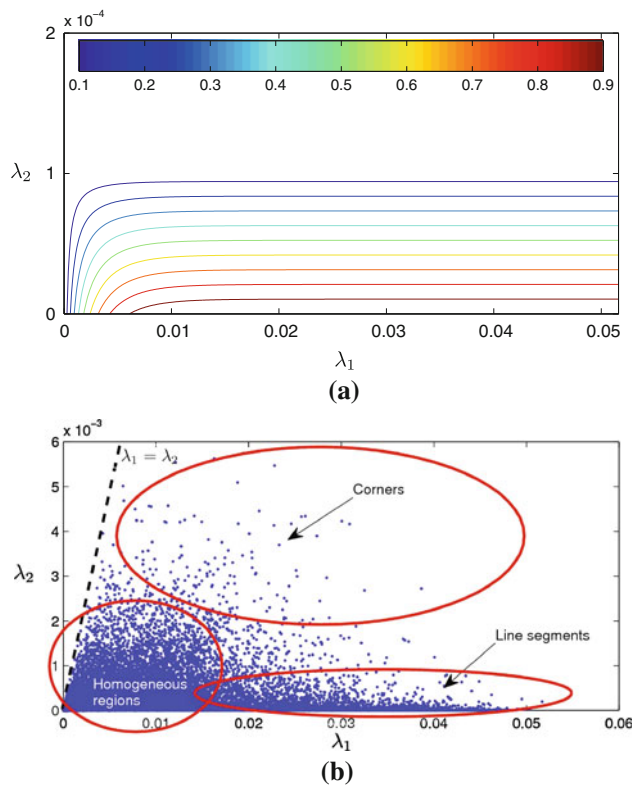


Fig. 5 The level sets of function $g(\lambda_1, \lambda_2)$ shown in **a**. In **b** a scatter plot of the set of pairs (λ_1, λ_2) for an example image. The concentration at the origin is typically due to the high number of pixels that belong to homogeneous regions of the image

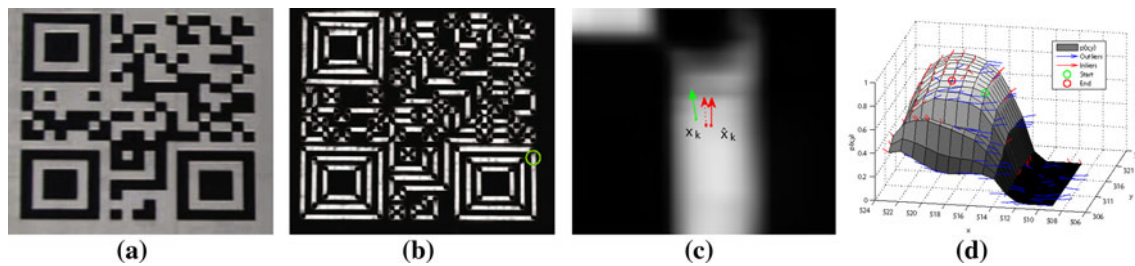


Fig. 6 The original image is shown in **a**, and the scaled pdf map in **b**. Note that corners receive low likelihood values and thus are depicted in *black*, as well as homogeneous regions. Only line segments have high likelihood values, painted in high bright levels; **c** shows a zoom

of the *highlighted* region (located at the *bottom right* part of the image shown in **b**). In **c**, the sample \mathbf{x}_k and the refined sample $\hat{\mathbf{x}}_k$ are shown. The *dotted* vector represents an intermediate iteration of wMS; finally **d** is the 3D representation of $\hat{p}(x, y)$ values of the considered region

4.3 Initialization and termination

The initialization of the algorithm is done by the iterative selection of pixels uniformly along the image until a pixel is found with $\hat{p}(x, y) > \mu$, where μ is the mean value of $\hat{p}(x, y)$ computed on the whole image. This initialization ensures a good candidate point for the line generator stage of the algorithm. The reason is that the histogram of $\hat{p}(x, y)$ typically follows a decreasing exponential distribution. Hence, most pixels of the image do not belong to line segments and form a mode close to zero. So, the selection of starting points above the mean ensures that we are not selecting points that do not likely belong to line segments. In the next steps, the slice sampler acts sequentially, finding automatically pixels with $\hat{p}(x, y)$ similar to the one of the initialization. Each time a line segment is generated, all the pixels that belong to it are removed from the set of pixels that can be selected by the slice sampler, as well as their neighbors in a window of $r \times r$ pixels, where r is the spatial bandwidth of MS; more details are given in next section.

A major virtue of this strategy is that typically only one candidate is required to generate a whole line segment, which marks as “visited” all its pixels so that the sampling strategy never draws a sample for that segment again. For that reason, it may happen that the slice sampler finds out that all the pixels surrounding the previous sample have been marked as visited, so that no more segments need to be detected in that region of the image. Therefore, the initialization criterion is applied again to produce a new start for the slice sampler, which will be referred to as a jump. Hence, the sampler allows to jump naturally from one region to another in the image according to the line segments generated in previous steps.

The proposed scheme sequentially selects the pixels of the image in order of importance, and allows to sweep all the pixels till the end. Nevertheless, it is also possible to determine a termination criteria according to the available computational resources, for example stopping when a maximum number of jumps is reached.

In summary, the sampling strategy sequentially generates samples $\mathbf{z}_k = (x_k, y_k)$, with an associated orientation $\theta_k = \arctan(I_y(x_k, y_k)/I_x(x_k, y_k))$. These samples are used by the line segment generation step to generate the line segments.

5 Line segment generation

For this stage of the algorithm, the sample \mathbf{z}_k and its associated normal orientation θ_k are used to compose the vector, or edge-point, $\mathbf{x}_k = (x_k, y_k, \theta_k)$, which will guide the line segment generation process. This process is composed of three independent stages, as shown in Fig. 1, which are applied on each new candidate \mathbf{x}_k . The point refinement step applies a multidimensional weighted mean shift procedure. It starts on \mathbf{x}_k and searches for a better representative edge-point of the line segment, denoted as $\hat{\mathbf{x}}_k = (\hat{x}, \hat{y}, \hat{\theta}_k)$. The line growing step searches for the two end points of the line segment under analysis. It is based on an oriented local growing algorithm that starts on $\hat{\mathbf{z}}_k$ governed by $\hat{\theta}_k$. The final step marks as visited those pixels that compose the line segment.

5.1 Refinement of the sample

The probabilistic nature of the slice sampling process delivers samples, \mathbf{x}_k , with high value of $p(x, y)$, that likely belong to a line segment. Nevertheless, in their neighborhoods there might be edge-points that better represent the position and orientation of the line segment. The refinement step searches for this representative, denoted as $\hat{\mathbf{x}}_k$, using a novel multidimensional wMS procedure, inspired on the well-known mean shift algorithm [8, 15]. MS is an iterative, non-parametric algorithm that can be used to find local maxima of an unknown density function from which a set of samples is available. At each iteration, MS operates on an starting sample and moves towards the most dense region of the search space in the neighborhood.

We propose to use MS with data samples weighted according to their likelihood values. For a given sample \mathbf{x}_k ,

its associated weighting factor is $\omega_k = \hat{p}(x, y)$. Hence, the wMS procedures find maxima following a twofold target. On the one hand, it looks for the maximum of $\hat{p}(x, y)$ within the neighborhood of \mathbf{z}_k , which is possible thanks to the weighting factors applied to the positions (x, y) , which are distributed uniformly along the image. The association of a weight to each pixel allows wMS to handle this spatial information as a non-uniform distribution. On the other hand, the orientation component of the search space makes that the wMS procedures move towards edge-points highly homogeneous in their orientations, which needs not to be at the maximum of $\hat{p}(x, y)$ in the neighborhood, but a better \mathbf{x}_k from which start the line growing stage.

Figure 6d shows a 3D representation of these values, corresponding to the zoomed region shown in (c): the elevation associated to each pixel corresponds to its line segment likelihood level, which is the weight applied during the wMS procedures. As observed, the pixels that belong to the line segment have higher weights, which are the target of the wMS procedures.

Mean shift works as a kernel-based non-parametric estimator of the pdf from which samples \mathbf{x}_k are drawn. Let us consider a set of samples, $\{\mathbf{x}_i\}_{i=1..N}$, from an unknown density $f(\mathbf{x})$. The multivariate weighted kernel estimator of $f(\mathbf{x})$ is defined as:

$$\hat{f}_\omega(\mathbf{x}) = \frac{1}{\sum_{i=1}^N \omega_i} \sum_{i=1}^N \omega_i K_H(\mathbf{x} - \mathbf{x}_i) \tag{3}$$

with the kernel defined as:

$$K_H(\mathbf{x}) = \frac{1}{|H|^{\frac{1}{2}}} K\left(\frac{\mathbf{x}}{H^{\frac{1}{2}}}\right) \tag{4}$$

where H is a symmetric positive definite 3×3 bandwidth matrix that specifies the “width“ of the kernel at each dimension. As a fully parameterized H increases the complexity of the estimation [8], the bandwidth matrix H is chosen for our approach as a diagonal matrix containing the corresponding bandwidths for each dimension: $H = \text{diag}[h_x^2, h_y^2, h_\theta^2]$. K is obtained from the product of symmetric univariate kernels:

$$K(\mathbf{x}) = \prod_{d=1}^3 c_d k(x_d) \tag{5}$$

where x_d are each component of the vector \mathbf{x} , c_d are the normalization constants, and the function k is defined as the Epanechnikov kernel [8]. Substituting (5) into (4), and the result into (3) yields:

$$\hat{f}_\omega(\mathbf{x}) = \frac{|H|^{-\frac{1}{2}} C}{\sum_{i=1}^N \omega_i} \sum_{i=1}^N \omega_i \prod_{d=1}^3 k\left(\left(\frac{x_d - x_{i,d}}{h_d}\right)^2\right) \tag{6}$$

where $x_{i,d}$ is the d th component of data sample \mathbf{x}_i and h_d is its corresponding bandwidth. Note that, as opposed to most approaches using MS, we are describing an estimator that considers not only a set of weighting factors for the kernels but also a multidimensional bandwidth matrix, which contains different bandwidth values for each dimension [15].

The modes in $f(\mathbf{x})$ that we are interested in, are located at the zeros of the gradient, $\nabla f(\mathbf{x})$. An estimator of the gradient of $f(\mathbf{x})$ is the gradient of $\hat{f}_\omega(\mathbf{x})$, that can be represented as follows:

$$\nabla \hat{f}_\omega(\mathbf{x}) = A \left[\sum_{i=1}^N \omega_i G(\mathbf{x} - \mathbf{x}_i) \right] \times \left[\left(\sum_{i=1}^N \omega_i G(\mathbf{x} - \mathbf{x}_i) \right)^{-1} \left(\sum_{i=1}^N \omega_i G(\mathbf{x} - \mathbf{x}_i) \mathbf{x}_i \right) - \mathbf{x} \right] \tag{7}$$

where A is a matrix gathering the constants and the bandwidth matrix:

$$A = \frac{2|H|^{-\frac{1}{2}} C}{\sum_{i=1}^N \omega_i} H^{-1} \tag{8}$$

and $G(\mathbf{x})$ is a diagonal 3×3 matrix defined as:

$$G(\mathbf{x}) = \text{diag} \left[-k' \left(\frac{x^2}{h_x^2} \right), -k' \left(\frac{y^2}{h_y^2} \right), -k' \left(\frac{\theta^2}{h_\theta^2} \right) \right] \tag{9}$$

The function k' is obtained as the differentiation of the kernel k , as defined in (5).

The last bracket in (7) is the mean shift vector, which represents the difference between the weighted mean of the data samples and the center of the kernel:

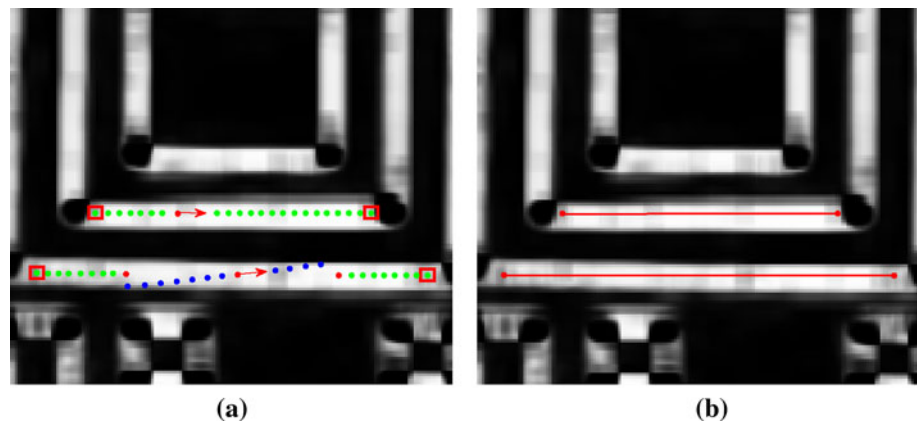
$$m_w(\mathbf{x}) = \frac{\left(\sum_{i=1}^N \omega_i G(\mathbf{x} - \mathbf{x}_i) \mathbf{x}_i \right) - \mathbf{x}}{\sum_{i=1}^N \omega_i G(\mathbf{x} - \mathbf{x}_i)} \tag{10}$$

The application of the wMS procedure generates mean shift vectors at each iteration towards the refinement of \mathbf{x}_k . An example is shown in Fig. 6c. The candidate vector-point given by the slice sampler is clearly near to a line segment, but the refined version, $\hat{\mathbf{x}}_k$, is much more accurate in position and orientation.

5.2 Line growing algorithm

The refined sample, $\hat{\mathbf{x}}_k = (\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$, is used as a starting point for the generation of the line segment. The line segment is delimited by its end points, $\mathbf{z}_a = (x_a, y_a)$ and $\mathbf{z}_b = (x_b, y_b)$, which are found using an oriented growing strategy inspired on the Bresenham algorithm [5]. This method is illustrated with the example upper line segment of Fig. 7a: starting from the edge-point (\hat{x}_k, \hat{y}_k) ,

Fig. 7 Oriented growing algorithm: **a** performance for two different growing cases, one of them with an accurate starting orientation (*top*), and the other (*bottom*) with a refinement step that corrects the orientation; **b** the obtained line segments for both cases are correct



corresponding to the arrow, the pixels along the orientation $\hat{\theta}_k$ (shown as dots) are connected while their orientations are in the range $\hat{\theta}_k \pm \Delta\theta$. In our experiments we have found that a good value for this parameter is $\Delta\theta = \frac{\pi}{8}$, which seems to work fine for a wide range of types of images, and it is also proposed by other authors that work with CCA techniques [7, 13]. The extreme points (marked with squares in the figure) of the obtained cluster are the target end-points of the line segment.

This procedure is applied iteratively until convergence is reached. For this purpose, an error measurement is defined for the set of points, indexed by i , covered by the line segment at iteration j , computed as follows:

$$\epsilon_i = \frac{1}{M} \sum_{i=1}^M |\theta_i - \hat{\theta}_j| \quad (11)$$

where θ_i is the orientation of each point of the line segment, $\hat{\theta}_j$ is the orientation that guides the line growing algorithm and M is the number of pixels connected by the growing algorithm. Note that the maximum error at each pixel is $|\theta_i - \hat{\theta}_j| \leq \Delta\theta$ (where $\Delta\theta = \frac{\pi}{8}$), hence $0 \leq \epsilon_i \leq \Delta\theta$. The process is considered to have converged when $\epsilon_j \geq \epsilon_{j-1}$, i.e., when the new iteration returns a set of pixels that does not fit better to the growing direction θ_j than the previous iteration. The value of $\hat{\theta}_j$ for each new iteration is obtained as the orientation of the segment that joins the refined end-points of the previous iteration. These points are refined as described in Sect. 5.1 by means of the application of one wMS procedure to each of them. The process is guaranteed to converge since the number of candidate end-points for a line segment is finite, and so are the orientations $\hat{\theta}_j$, so that there is a global minimum error ϵ_j . Nevertheless, we have seen in our experiments that, in average, the process converges in two or three iterations.

The reason of using this iterative strategy is to compensate potential small deviations between $\hat{\theta}_k$ and the actual line segment orientation. An example is depicted in

the lower segment of Fig. 7a. As shown, the application of two new wMS procedures, one on each end-point restarts the growing algorithm to find a more accurate line segment, achieving convergence at the second iteration. Fig. 7b shows the obtained segments for the examples shown in (a).

6 Complexity of the algorithm

In this section we provide an estimate of the complexity of the whole algorithm. For this purpose we would assume that sums, products, divisions and the rest of basic operations are equally costly (as done by other authors of related works [14]).

The number of operations of the proposed method is approximately proportional to the number of pixels of the image, N . The most time consuming part of the algorithm is the computation of the estimate of the likelihood function $\hat{p}(x, y)$, since it involves the calculation of the eigenvalues and vectors for all the pixels of the image. The operations that are carried out are the calculation of the Sobel approximation to the spatial derivatives $O(16N)$, the computation of the gradient orientation $O(2N)$, the obtention of the eigenvalues $O(10N)$ and the computation of μ , $O(N)$. The result is $O(29N)$ which is proportional to the size of the image.

The rest of computations are done for each line segment and consume much less processing resources. The Bresenham algorithm plus the wMS-based refinement is $O(L + 3R)$, where L is the average length of a line segment, and $R = r \times r$ is the squared spatial bandwidth of wMS ($r = h_x = h_y$). $O(L)$ refers to the computation of the position of each candidate pixel in the growing strategy, and $O(3R)$ corresponds to the execution of 3 MS procedures, one for setting the starting point of the growing strategy, and one for each end-point to refine its position. Note that $M(L + 3R) \ll 29N$ for average values of $L = 50$ pixels, $M = 400$ line segments and $N = 600 \times 400$ pixels.

Table 1 Average process time (in milliseconds) for different image sizes and requested line segments

Image Size	Requested number of line segments			
	50	100	500	∞
180×144	16	16	16	16
320×240	31	31	31	31
360×288	46	47	47	47
640×480	125	125	156	156
$1,024 \times 720$	281	281	313	390
$1,366 \times 768$	391	391	422	563
$1,280 \times 1,024$	485	484	531	688
$1,680 \times 1,050$	641	656	687	922
$1,600 \times 1,200$	703	703	735	1,000
$1,900 \times 1,200$	812	844	846	1,172
$3,072 \times 2,304$	2,578	2,578	2,609	3,781

This theoretical analysis is reflected in the results of next sections, especially in Table 1, where the processing time of the algorithm is evaluated for the detection of different number of line segments.

7 Tests and results

This section includes the description of different tests applied on the developed SSWMS algorithm, which are focused on demonstrating its performance in terms of speed, accuracy, and flexibility. A comparison with other related state-of-the-art line detectors is also presented.

7.1 Performance analysis

For the tests, we have implemented our solution in C++ programming language using OpenCV v1.1 libraries, and we have run the tests on a Core2Duo processor at 2.2 GHz. We have collected and processed a large set of images of several environments, such as buildings, roads, facades, portraits, etc., with different formats, sizes and levels of noise. Figure 8 shows some of these images and their associated detected line segments. Observe that most of significant line segments in the images are correctly detected, achieving great accuracy and resulting in a very reduced number of false negatives or misdetections. Furthermore, it is also remarkable that the algorithm generates very few false positives in areas of the image that contain edges but without actual line segments, such as the leaves of the trees, the mountains, the hair or the small elements at the far distance. Ordered from left to right and from top to bottom, the third image shows a very interesting property of the SSWMS thanks to the sequential sampling step: it provides satisfactory results for all the straight edges of the

image even though there are some (the ones corresponding to the shadows) that are much stronger than the rest of edges in the image. This would cause problems in methods that process sequentially based on sorting the edge points of the image according to their magnitude.

Regarding processing times, the algorithm was designed to perform in real time for medium size images, achieving an average of 125 ms (8 fps) for images about 640×480 pixels and below 50 ms (20 fps) for 360×288 pixels, which are typical image sizes for most online video processing applications. The processing time increases linearly with the size of the images, so that larger ones, with HD resolutions such as $1,280 \times 720$, $1,440 \times 1,080$ or $1,920 \times 1,080$ require processing times of 430 ms (2.32 fps), 650 (1.5 fps), and 1,000 ms (1 fps), respectively.

7.2 Discussion about parametrization

It must be considered that the processing times presented before correspond to the execution of the SSWMS until it has searched for line segments along the whole image. The sequential and probabilistic nature of the algorithm allows to stop computing when a requested number of line segments has been obtained. Typically, for most computer vision applications, only the most representative line segments are of utility. Table 1 shows the time consumed by the SSWMS for the detection of different number of line segments for different image sizes. As shown, the processing times are very reduced, and perfectly suitable for real-time computer vision applications. On the one hand, the computational time required for images with resolutions below 640×480 do not significantly vary for different target numbers of line segments, as the most consuming part of the algorithm is the computation of the eigenvectors. Therefore, for medium and small images, it is worthy to apply the full search, with no parameters, as the gain of time is negligible. On the other hand, for larger image sizes, the reduction of time when reducing the number of requested line segments is more significant. For instance, the application of the SSWMS on an image with $1,600 \times 1,200$ pixels, would spend near 1,000 ms, while applying the SSWMS limiting the search to the first 500 line segments, would result in a reduction of processing time in about 30%. In any case, detected line segments are those ones having more intense values of $p(x, y)$, which are typically the most significant lines in the image.

The only parameter of the algorithm that has to be fixed, as mentioned along the paper, is the spatial component of the bandwidth vector of the mean shift procedures, r . By default it is fixed to $r = 3$, which makes the system perform excellent for medium-size images. Nevertheless, an even better performance can be obtained by automatically adapting this value according to the size of the images. The

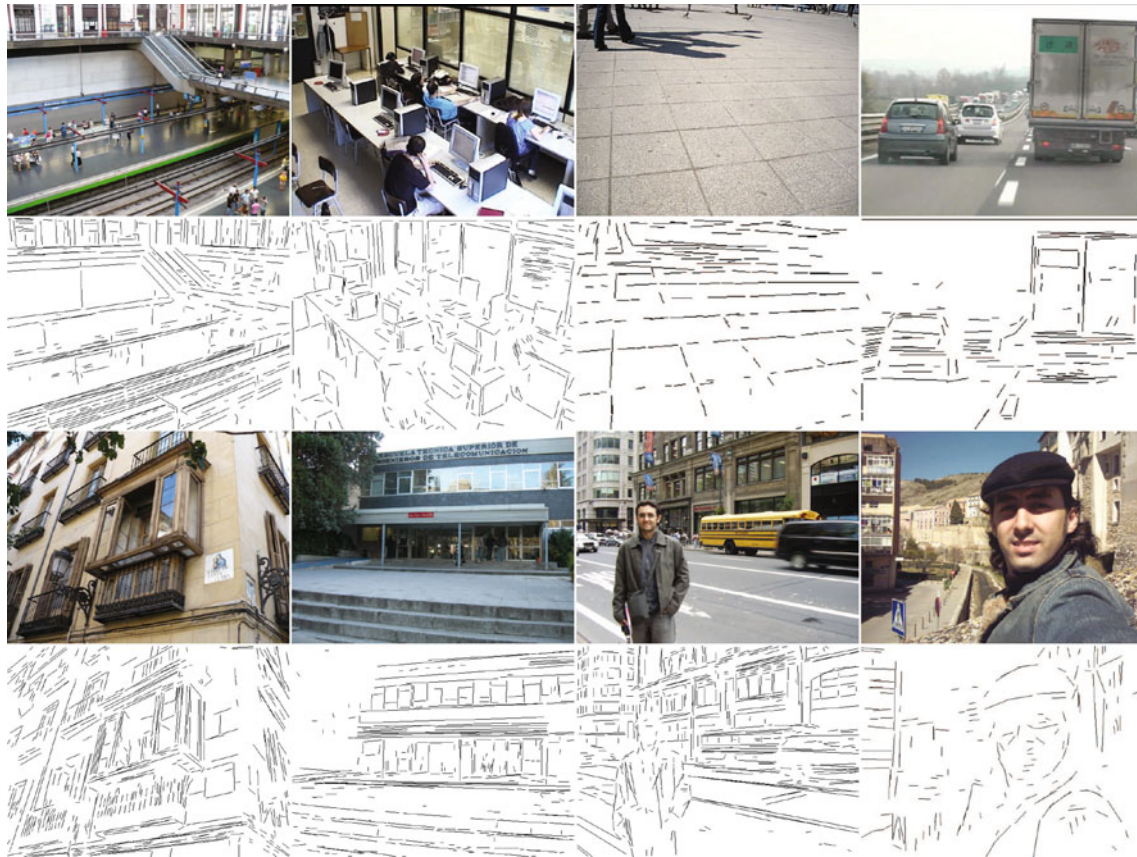


Fig. 8 Some results of the SSWMS applied on different real images

bandwidth is an indicator of the resolution of the detector, i.e., the minimum distance between two detected line segments. For that reason, as the images increase in size better results are obtained increasing the size of this parameter. The results for an example image using different values of r is shown in Fig. 9. As shown, for small values of r , the number of detected line segments is much higher. Increasing this parameter makes the system work towards the detection of dominant line segments. For instance, for $r = 9$, the number of detected line segments is 75, which mostly correspond to the main edges of the building.

7.3 Comparison with other methods

We have selected two line segment detection algorithms to compare with our method: the LSD (Line Segment Detector [13]) and the PPHT (Progressive Probabilistic Hough Transform [11]) for which efficient implementations are available (the authors of LSD offer their implementation in their website, while the PPHT is a very well-known algorithm that has been efficiently implemented as a function inside the OpenCV 1.1 libraries). The motivation of this selection is that this implementation of the PPHT is a very

good representative of Hough-based methods as well as, in our knowledge, it is the fastest method in the literature. The LSD has been recently published and it offers linear-time operation and very accurate results, and is selected for comparison as it is the best algorithm derived from the Burns method [6].

The first test was conducted to check the performance of these algorithms in the presence of noise. An example image, with significant added noise and the detected line segments obtained with the different methods is shown in Fig. 10. The upper row of this figure illustrates the robustness of the SSWMS against noise: the LSD is unable to detect segments, the PPHT produces hundreds of noisy line segments, while our approach detects, partially splitted, the most important line segments in the image. In the second row, a gaussian filter is used to remove partially the noise, as suggested by [13] to improve its accuracy in the presence of noise. In this situation, both SSWMS and LSD show similar performance, while the PPHT still generates too many noisy segments.

Figure 11 shows a scatterplot of the processing time of our approach against the size of the image (in pixels) compared to the PPHT and LSD algorithms: (a) shows the comparison for small and medium size images, and (b) for

Fig. 9 Results of the SSWMS line segment detection using different values of the mean shift spatial component of the bandwidth, r

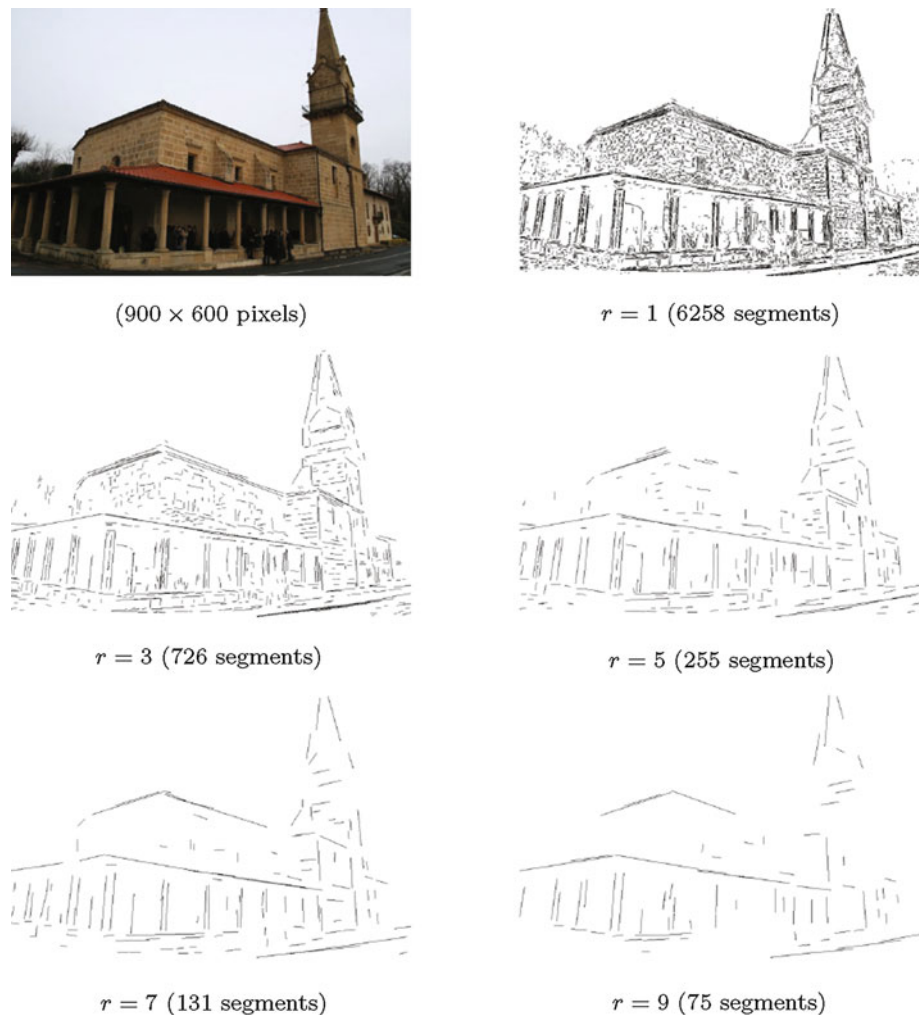
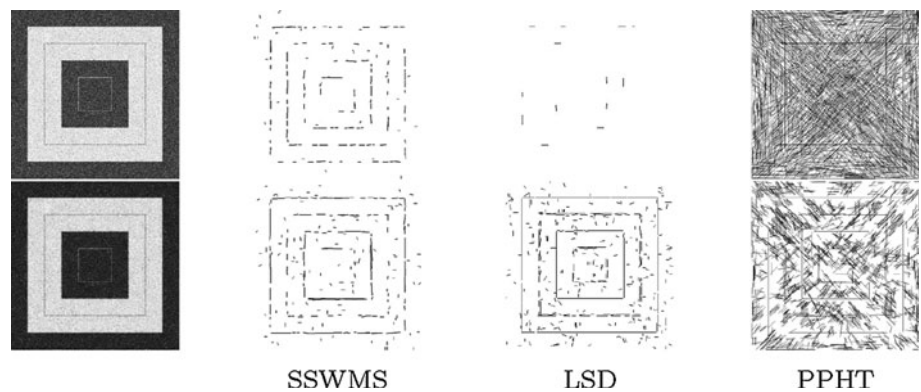


Fig. 10 Comparison of the performance of the compared methods against noise. The upper row shows the results obtained from the noisy image; and the second row shows the results for the smoothed version of the image



large images. The PPHT is on average 10% faster than SSWMS considering the whole set of images used for the comparisons (more than 200 images), while the LSD is between 10% (for large images) and 30% (for small and medium size images) slower than our approach. Note that for the comparison we had to tune the parameters of the PPHT for each image independently to obtain the best results in terms of speed and accuracy (such as the

resolution of the transform space, the minimum vote threshold, and minimum length of the line segments), while the SSWMS needs no parameter tuning. The accuracy of these methods is illustrated, for different images in Fig. 12. As shown, the PPHT shows the higher number of false detections and missdetections. The LSD offers very good results for almost all images, specially those that have high-contrast edges. Nevertheless, for some images, such

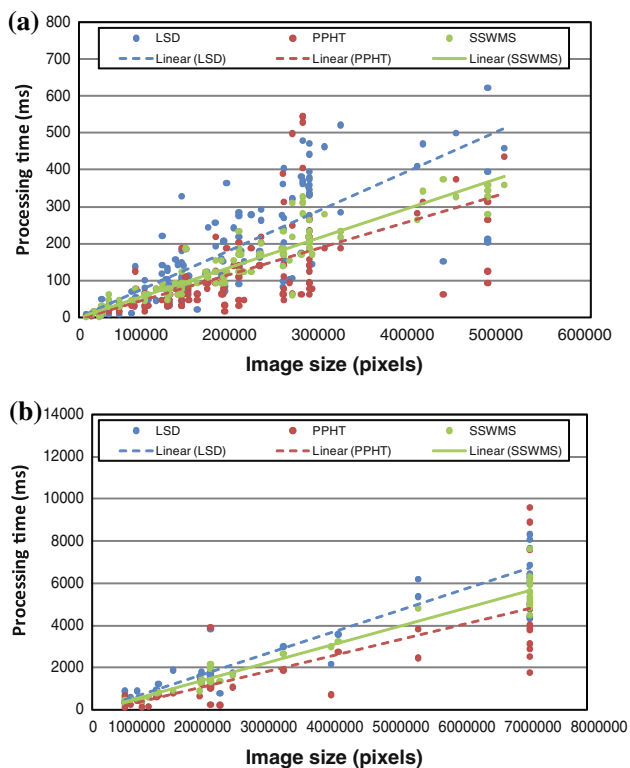


Fig. 11 Process time of the SSWMS, PPHT and LSD methods for more than 200 images with different sizes: **a** images up to about 600×900 pixels; and **b** larger images up to $3,702 \times 2,304$ pixels. The lines represent the linear tendency of the data along the image size axis (solid lines represents our method). As shown, the faster method is the PPHT, while the SSWMS shows higher performance than LSD in both cases

as the one showing the ground (first row in Fig. 12), it fails to detect important line segments, and it produces too many line segments, retrieving multiple replied segments that add no information about the scene. In all these cases, the SSWMS shows excellent performance, detecting the most important line segments with a highly reduced number of false detections and misdetections.

7.4 Recall and precision

This section describes the comparison of the performance of the proposed algorithm with other methods in terms of recall and precision. Recall is related to the number of misdetections, and precision to the number of false alarms. These evaluation metrics can be computed as:

$$\text{Recall (\%)} = \frac{\# \text{ Correct detections}}{\# \text{ Ground truth line segments}} \quad (12)$$

$$\text{Precision (\%)} = \frac{\# \text{ Correct detections}}{\# \text{ Total detections}} \quad (13)$$

In this context, a detected line segment is considered to be a correct detection if it is similar to an existing ground truth line segment in terms of relative orientation, distance

between mid-points and length. The target of the test is to evaluate, with objective results, the detection results on different situations, which correspond to typical conditions of real images: (1) variable size, (2) noise, and (3) perspective distortion. Modifying the size of the images helps to evaluate the response of the methods against variations of the length of the line segments. The introduction of noise shows how flexible the algorithms are in providing good results when the spatial coherence of pixel information (such as gradients) is lost or corrupted due to the noise. Finally, the perspective distortion illustrates another important property: this type of transformation implies a non-homogeneous modification of the image area. Hence, the length of the line segments is non-uniformly modified. Therefore, this test illustrates the ability of the methods to compute long and short line segments in the same image.

The RP values have been obtained for an example image, shown in Fig. 13a, and for a number of its modifications, illustrated in Fig. 13b, c, which follow the above-mentioned variations in size, noise and perspective distortion. The recall and precision results are shown in Fig. 14, separately for the three types of modifications, where the values corresponding to the original image are shown with a thick black marker in the three graphs². The graph showing “variable size” depicts the RP values obtained increasing the size of the image; “variable noise” stands for the addition of Gaussian noise with standard deviation values ranging from 0 to 0.07 in 0.01 steps. An example image with noise is shown in Fig. 13b. Finally, “variable perspective distortion” illustrates the results obtained by applying perspective transformations on the image, such as the one shown in Fig. 13c.

The results related to the modification of the size of the image indicates that the SSWMS renders very accurate results with very low misdetections and false alarms, while the performance of the LSD decreases as it provides more false alarms as the size of the image increases. The recall of the three methods increases since it is easier to detect all the existing line segments as the image increases.

The addition of noise, as already commented in the example of Fig. 10, is very harmful for all methods. Therefore, RP curves tend to decrease both the recall and precision values, although to a lesser extent for the SSWMS algorithm, which still shows RP above 50/50 for added noise with 0.05 standard deviation.

The results corresponding to the images with perspective distortion show that the recall values decrease as the number

² SSWMS RP, 91.73/95.35; LSD RP, 90.35/73.33; PPHT, RP: 38.19/89.91. These numbers mean that both the LSD and SSWMS offer great results for this image, although the LSD delivers more false alarms. The PPHT suffer more misdetections and thus its recall value is very low.

Fig. 12 Comparisons of the line segments obtained for different images applying different methods. From left to right: SSWMS, LSD and PPHT

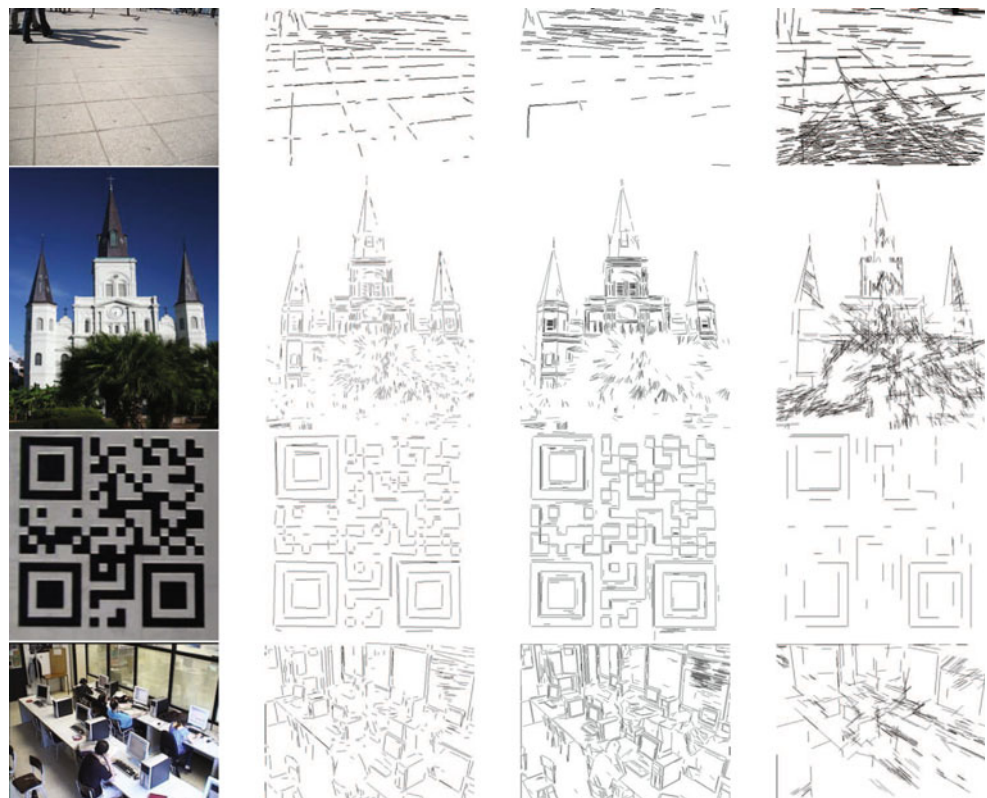


Fig. 13 Example image used to test recall and precision: **a** original image with ground truth line segments; **b** image with added gaussian noise; and **c** image with perspective distortion

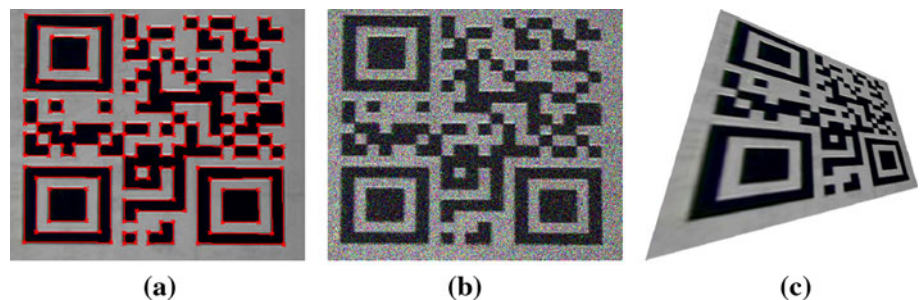
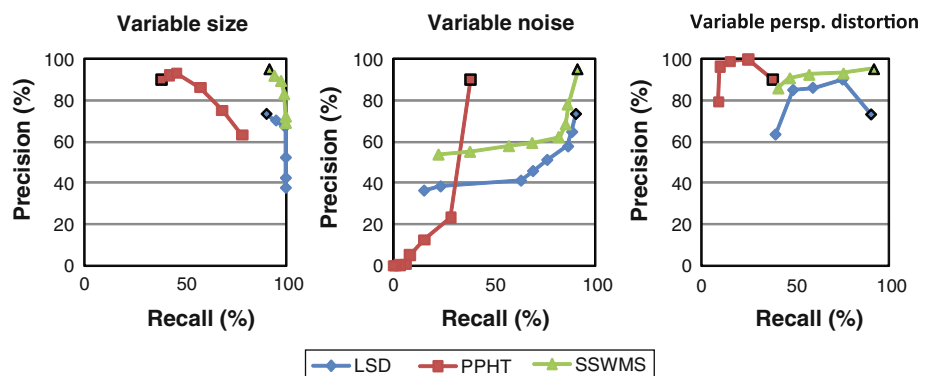


Fig. 14 Recall and precision graphs for different situations: varying size, adding noise and perspective distortion for the three methods. The starting point is marked with a thick black edge, and is the same for the three graphs



of line segments whose length is reduced grows. Nevertheless, the SSWMS algorithm still keeps a very good value of RP especially compared with the PPHT, which is not able to detect a large number of line segments due to the distortion.

8 Example application

As commented in Sect. 1, line segments are useful image features for a number of applications in the computer

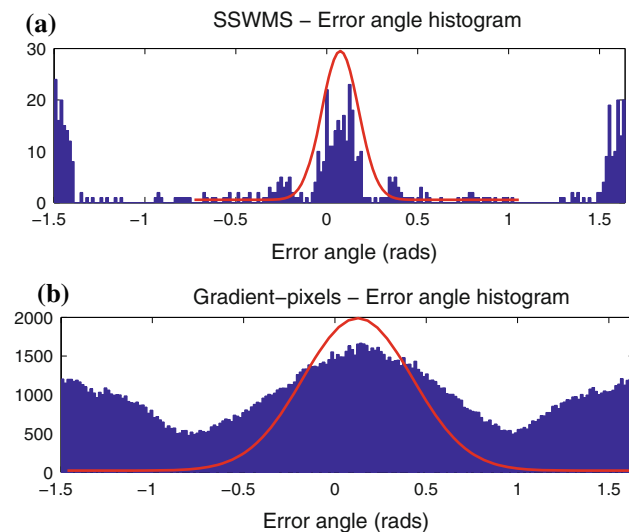


Fig. 15 Error angle histograms: **a** line segments obtained with SSWMS; and **b** gradient-pixels

vision field. As an example, we have successfully used the proposed SSWMS algorithm for the computation of vanishing points in sequences of images. The accuracy of the SSWMS has been evaluated computing the orientation error of the obtained line segments with respect vanishing points following the approach by Košecká and Zhang [20].

For this purpose we have used the York Urban Data Base (YUDB) [10], which is a database of 102 images of man-made structured environments with ground truth vanishing points (which correspond to the three main orthogonal directions of the scene). For each image of the YUDB we have run the SSWMS and we have evaluated the orientation error of the obtained line segments with respect to each vanishing point. Figure 15a shows an example histogram of the error of the detected line segments with respect to one vanishing point. The mode around zero correspond to the error of the line segments actually meeting at the vanishing point. The rest of the histogram corresponds to line segments not meeting that vanishing point. As shown, the peak can be fitted as a normal distribution with a mean close to zero and a low variance. The error of the line segments obtained with the SSWMS is compared with the actual error of the image information, which can be characterized by the gradient vectors computed for each pixel of the image. The corresponding histogram of these gradient vectors is shown in Fig. 15b. As can be observed, the error distribution is more peaked for the line segments, which illustrates that the use of SSWMS reduces the error in the orientation and thus increases the accuracy of the information that is used to compute vanishing points. This experiment is repeated for all the vanishing points of all the images of the YUDB,

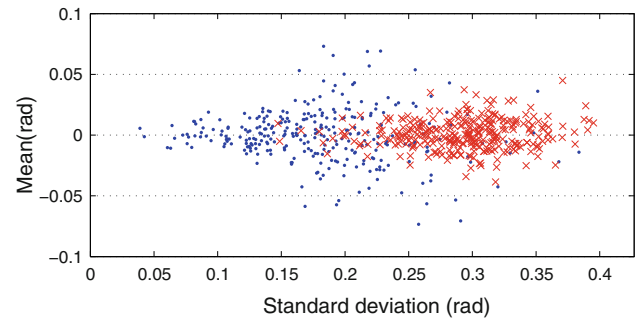


Fig. 16 Scatter plot of the mean and standard deviation of the normal fit of the inliers mode of the error histogram. The results of the normal fit of the line segments obtained with SSWMS is shown in *dots*. The normal fit corresponding to gradient-pixels is shown with *crosses*. Realize that there are 306 data points, since each image of the YUDB contains three vanishing points

obtaining mean and standard deviation values. The result is illustrated in Fig. 16, which shows a scatter plot of the obtained statistics, showing with *dots* the results for the SSWMS line segments, and with *crosses* the results of the gradient-pixels. As can be observed, the distribution of error of the SSWMS is more concentrated on zero mean values, with lower standard deviation.

9 Conclusions

In this paper, we have proposed a novel approach for accurate and fast detection of line segments in images, specially devoted for real-time vision applications. It has been designed to adapt itself to the characteristics of the images; hence, there is no need to tune any input parameter. It is based on a sequential sampling strategy, which uses the slice sampler to draw pixels in the image that likely belong to line segments. At each drawn sample, an iterative line growing strategy, based on mean shift is applied which finally finds the end-points of the line segment.

The outstanding performance of this strategy has been demonstrated in terms of accuracy, flexibility and, mainly, speed. A comparison with other line segment detection methods has been carried out. It has shown that our method offers an excellent trade-off between very accurate methods and fast ones. One of the major abilities of the proposed algorithm is that it finds the most important line segments for any type of images, including noisy ones, without the need of tuning any parameter.

Acknowledgments This work has been partially supported by the Ministerio de Ciencia e Innovación of the Spanish Government under project TEC2007-67764 (SmartVision), and by the Comunidad de Madrid under project S-0505/TIC-0223 (Pro-Multidis).

References

1. Aggarwal N, Karl WC (2006) Line detection in images through regularized Hough transform. *IEEE Trans Image Process* 15(3):582–591
2. Almansa A, Desolneux A, Vamech S (2003) Vanishing point detection without any a priori information. *IEEE Trans Pattern Anal Mach Intell* 25(4):502–507
3. Bandera A, Perez Lorenzo JM, Bandera JP, Sandoval F (2006) Mean shift based clustering of Hough domain for fast line segment detection. *Pattern Recognit Lett* 27(6):578–586
4. Bishop CM (2006) *Pattern recognition and machine learning (Information Science and Statistics)*. Springer
5. Bresenham J (1965) Algorithm for computer control of a digital plotter. *IBM Syst J* 4(1):25–30
6. Burns JB, Hanson AR, Riseman EM (1986) Extracting straight lines. *IEEE Trans Pattern Anal Mach Intell* 8(4):425–455
7. Canny J (1986) A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell* 8(6):679–698
8. Comaniciu D, Meer P (2002) Mean shift: A robust approach toward feature space analysis. *IEEE Trans Pattern Anal Mach Intell* 24:603–619
9. Dayhot R (2009) Statistical Hough transform. *IEEE Trans Pattern Anal Mach Intell* 31(8):1502–1509
10. Denis P, Elder JH, Estrada FJ (2008) Efficient edge-based methods for estimating Manhattan Frames in urban imagery. In: *IEEE Proc European Conference on Computer Vision* 2:197–210
11. Galambos C, Kittler J, Matas J (1999) Progressive probabilistic hough transform for line detection. *IEEE Comput Soc Conf Comput Vis Pattern Recognit* 1:1554
12. Gilks W, Richardson S, Spiegelhalter D (1996) *Markov Chain Monte Carlo Methods in practice*. Chapman and Hall/CRC
13. Grompone von Gioi R, Jakubowicz J, Morel JM, Randall G (2009) LSD: a fast line segment detector with a false detection control. *IEEE Trans Pattern Anal Mach Intell* 99(1)
14. Guru DS, Shekar BH, Nagabhushan P (2004) A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognit Lett* 25(1):1–13
15. Han B, Comaniciu D, Zhu Y, Davis LS (2008) Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Trans Pattern Anal Mach Intell* 30(7):1186–1197
16. Heath M, Sarkar S, Sanocki T, Bowyer K (1998) Comparison of edge detectors: a methodology and initial study. *Comput Vis Image understand* 69(1):38–54
17. Kälviäinen H, Hirvonen P, Xu L, Oja E (1994) Comparisons of probabilistic and non-probabilistic Hough transforms. *Proceedings of 3rd European conference on computer vision*, pp 351–360
18. Khan P, Kitchen L, Riseman EM (1990) A fast line finder for vision-guided robot navigation. *IEEE Trans Pattern Anal Mach Intell* 12(11):1098–1102
19. Kiryati N, Eldar Y, Bruckstein AM A probabilistic hough transform. *Pattern Recognit* 24(4):303–316
20. Košecká J, Zhang W (2003) Video compass. *European conference on computer vision, LNCS 2350*, Springer, pp 476–491
21. Liebowitz D, Zisserman A (1998) Metric rectification for perspective images of planes. In: *IEEE Proc. computer vision and pattern recognition* 0:482–488
22. Liebowitz D, Criminisi A, Zisserman A (1999) Creating architectural models from images. *Comput Graphics Forum* 18(3):39–50
23. Maduro C, Batista K, Peixoto P, Batista J (2008) Estimation of vehicle velocity and traffic intensity using rectified images. *IEEE international conference on image processing*, pp 777–780
24. Nacken PFM (1993) A metric for line segments. *IEEE Trans Pattern Anal Mach Intell* 15:1312–1318
25. Neal R (2000) Slice sampling. *Ann Stat* 31:705–767
26. Nevatia R, Babu KR (1980) Linear feature extraction and description. *Comput Graphics Image Process* 13:257–269
27. Pflugfelder R (2008) *Self-calibrating cameras in video surveillance*. PhD thesis, Graz University of Technology
28. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. *Eur Conf Comput Vis* 1:430–443
29. Serby D, Meier EK, Van Gool L (2004) Probabilistic object tracking using multiple features. In: *Proceedings of the 17th international conference on pattern recognition* 2:184–187
30. Sirakov NM, Kojouharov H, Sirakova NN (2010) Tracking Neutrophils cells by active contours with coherence and boundary improvement filter. In: *IEEE Proc. SSIAI2010* 5–8
31. Stephens RS (1991) Probabilistic approach to the hough transform. *Image Vis Comput* 9(1):66–71
32. Walsh D, Raftery AE (2002) Accurate and efficient curve detection in images: the importance sampling Hough transform. *Pattern Recognit* 35:1421–1431
33. Wang H, Cheng Y, Fang T, Tyan J, Ahuja N (2006) Gradient adaptive image restoration and enhancement. In: *IEEE Proc. international conference on image processing*, pp 2893–2896
34. Xu L, Oja E, Kultanen P (1990) A new curve detection method: Randomized Hough transform (RHT). *Pattern Recognit Lett* 11(5):331–338
35. Yuen SYK, Lam TSL, Leung NKD (1993) Connective Hough transform. *Image Vis Comput* 11:295–301