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

Article *in* Pattern Analysis and Applications · May 2011

DOI: 10.1007/s10044-011-0211-4 · Source: DBLP

Marcos Nieto
Vicomtech
60 PUBLICATIONS **661** CITATIONS
SEE PROFILE

Carlos Cuevas
Universidad Politécnica de Madrid
41 PUBLICATIONS **228** CITATIONS
SEE PROFILE

Luis Salgado
Universidad Politécnica de Madrid
125 PUBLICATIONS **1,227** CITATIONS
SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Line segment detection using weighted Mean Shift procedures on a 2D Slice sampling strategy

Marcos Nieto · Carlos Cuevas · Luis Salgado · Narciso García

Received: date / Accepted: date

Abstract A new approach towards accurate line segment detection in images 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 to tune input parameters. Although many works have been presented about 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 [28], plane rectification techniques [21,23], 3D reconstruction [22] or object 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 subjacent 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, the results section 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

Marcos Nieto, Carlos Cuevas, Luis Salgado, and Narciso García
Grupo de Tratamiento de Imágenes
Universidad Politécnica de Madrid
E28040 - Madrid, Spain
E-mail: {mnd, ccr, L.Salgado, narciso}@gti.ssr.upm.es

of structured environments [20,27], which require real-time operation while 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 i at location (x_k, y_k) the module and orientation of the gradient can also be estimated as described in [10].

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) [1,3,10,12,17,19,31]; and pixel clustering strategies in the image plane [6,14,15,18].

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]. Some other works just face the problem of finding lines, as done by [31,34,19] or recently Aggarwal with the Regularized HT [1].

The major drawback of the works based on the HT is the excessive algorithm complexity [31,10], which avoids its use on online applications.

Random sampling applied to the HT 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-steps 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 Banderas 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 (missdetections due to the termination criteria), especially in very fast detectors as the PPHT (Progressive Probabilistic HT) by Matas et al. [12]. Besides, they require a large set of application-dependant 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 [18,14]. 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 [35] proposes a connected version of the HT, which selects at random an edge pixel and then apply a 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 [15, 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 [29], 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 shares some properties with methods of the two abovementioned groups, although it can not be classified as belonging to any of them. On the one hand, it uses a sequential sampling strategy, which is an

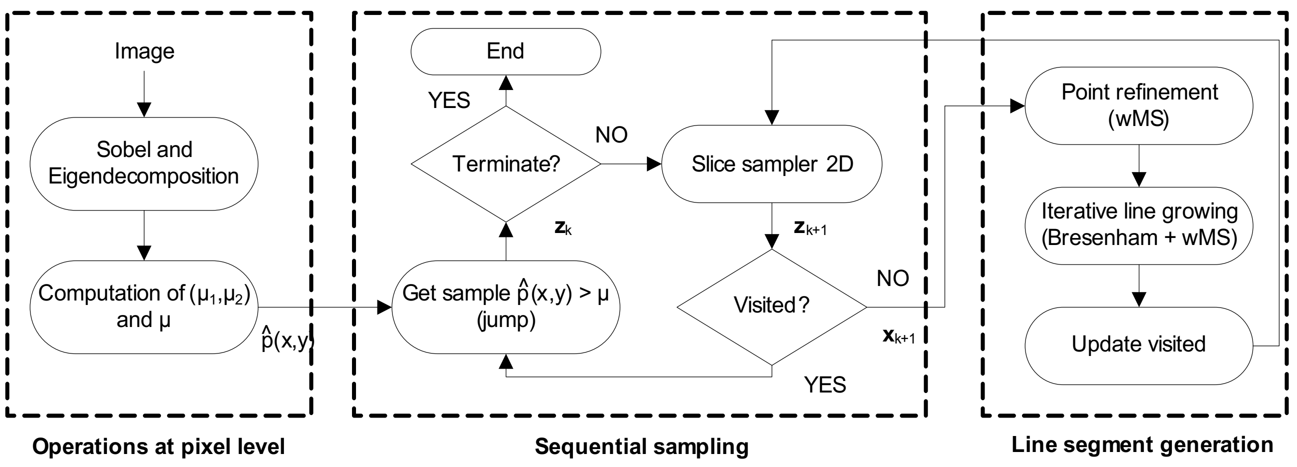


Image
Sobel and Eigendecomposition
Computation of (μ1,μ2) and μ
p̂(x,y)
Operations at pixel level
End
YES
Terminate?
NO
Slice sampler 2D
zk
zk+1
Get sample p̂(x,y) > μ (jump)
Visited?
NO
xk+1
YES
Sequential sampling
Point refinement (wMS)
Iterative line growing (Bresenham + wMS)
Update visited
Line segment generation

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, $\mathbf{z}_k$. 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.

improved version of random sampling approaches based on the HT. On the other hand, it exploits the eigenvalues and vectors information as also done by some pixel clustering methods, although in a novel way by the computation of the 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: (i) they have significant gradient magnitude; and (ii) 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, which is based on the slice sampling algorithm [4, 25], sequentially selects pixels in the image, denoted as $\mathbf{z}_k = (x_k, y_k)$, that are good candidates to belong to line segments, according to the computed likelihood $\hat{p}(\mathbf{z}_k)$. The design of this stage ensures that no repeated samples are selected, so that $\mathbf{z}_k \neq \mathbf{z}_c \; \forall c < k$. If the slice sampler proposes a sample $\mathbf{z}_k$ 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 $\mathbf{x}_k = (x_k, y_k, \theta_k)^1$. 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 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.

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

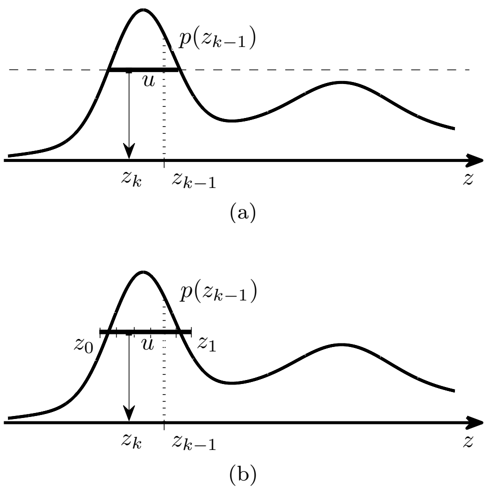


Fig. 2 Univariate slice sampling: (a) the uniform u value determines the slice through $p(z)$; and (b) the practical implementation uses fixed length steps to determine the range in which z is uniformly sampled.

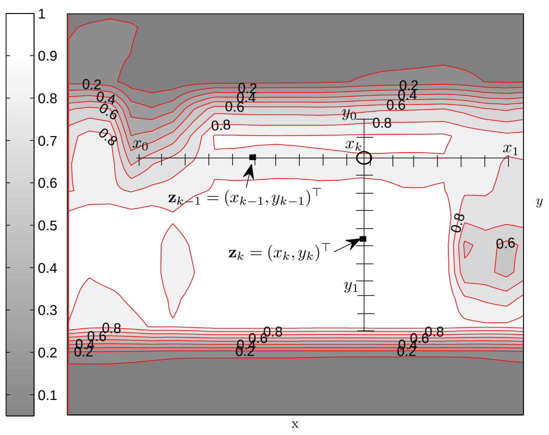


Fig. 3 The level set plot allows to observe that $\mathbf{z}_{k-1}$ is located in a region with levels between 0.6 and 0.8. The slice procedure on x produces the region limited by x_0 and x_1. The selected sample x_k is fixed and the region between y_0 and y_1 is obtained for the y dimension. The final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

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.

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 [13]. 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 behave 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. 2 (a). 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. 2 (b). 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. Fig. 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 an efficient 2D sampling. As a remark, note that if the order of the one-dimensional samplers is reversed, a different fi-

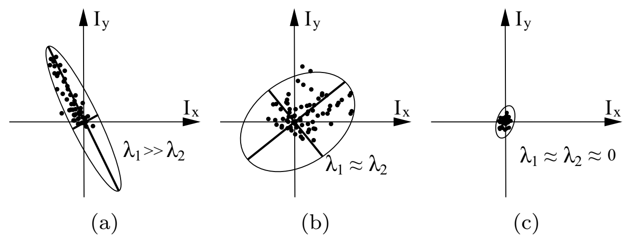


Fig. 4 Eigenvalues illustrating gradient structures. Black dots are the (I_x, I_y) values of the neighbor pixels of an example pixel that corresponds to: (a) a line segment; (b) a corner or an heterogeneous gradient region; and (c) homogeneous region.

nal 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.

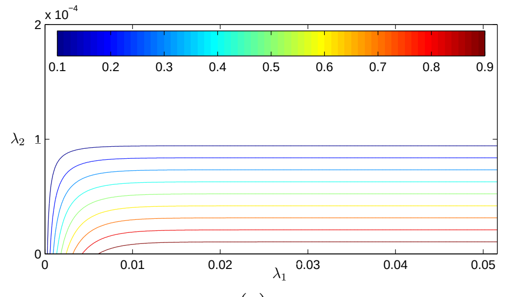
4.2 Line segment likelihood

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.

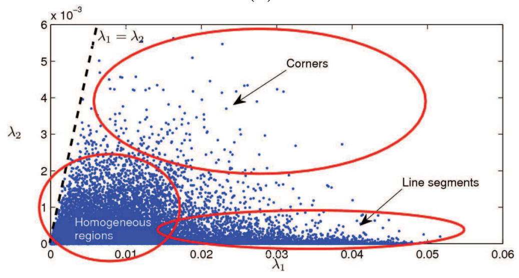
Different methods have been addressed in the literature to characterize image pixels to belong to corners, homogeneous regions or line segments. Most of them use the covariance matrix or image tensor matrix for each pixel and their associated eigenvalues, (λ_1, λ_2) with $\lambda_1 > \lambda_2$, and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

The eigenvalues illustrate the dispersion of the gradient along their associated eigenvectors [29] 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 satisfy 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:



(a)



(b)

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.

$$
\begin{aligned}
p = g \circ f : I &\rightarrow (\lambda_1, \lambda_2) \rightarrow \mathbb{R} \\
(x, y) &\mapsto \hat{p}(x, y) = g(f(x, y))
\end{aligned}
\tag{1}
$$

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:

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

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 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. 5 (a), while the associated scatter plot of (λ_1, λ_2) is shown in Fig. 5 (b).

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.

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 $p(x, y)$ 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 the 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, 9]. 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

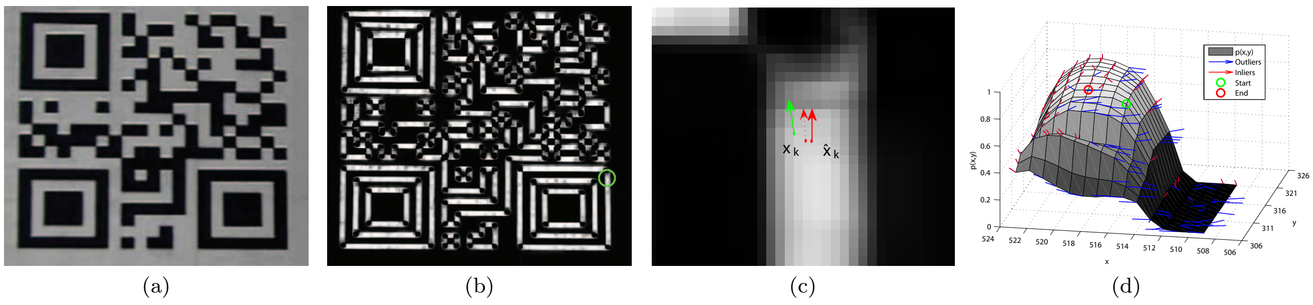


Fig. 6 The original image in shown in (a), and the scaled pdf map in (b). Note that corners receive low likelihood values and thus are depicted in black, as well as homogeneous regions. Only line segments has high likelihood values, painted in high bright levels; (c) shows a zoom of the highlighted region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

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.

Fig. 6 (d) 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:

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

with the kernel defined as:

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

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: $\mathrm{H} = diag[h_x^2, h_y^2, h_\theta^2]$. 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:

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

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 [9].

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 $\widehat{f}_\omega(\mathbf{x})$, that can be represented as follows:

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

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

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

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

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

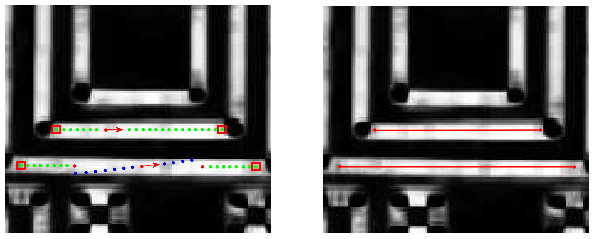


Fig. 7 Oriented growing algorithm: (a) performance for two different growing cases, one of them with an accurate starting orientation (top), and the other (bottom) with a refinement step that corrects the orientation; (b) the obtained line segments for both cases are correct.

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. 6 (c). 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. 7 (a): starting from the edge-point $(\hat{x}_k, \hat{y}_k)$, 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 the best value for this parameter is $\Delta\theta = \frac{\pi}{8}$, which is a good choice for all kind of images, and it is also proposed by other authors that work with CCA techniques [7,14]. 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:

$$\epsilon_i = \frac{1}{M} \sum_{i=1}^{M} \mid \theta_i - \hat{\theta}_j \mid \tag{11}$$

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 $\mid \theta_i - \hat{\theta}_j \mid \leq \Delta\theta$, 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 section 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. 7 (a). 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. 7 (b) 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 [15]).

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

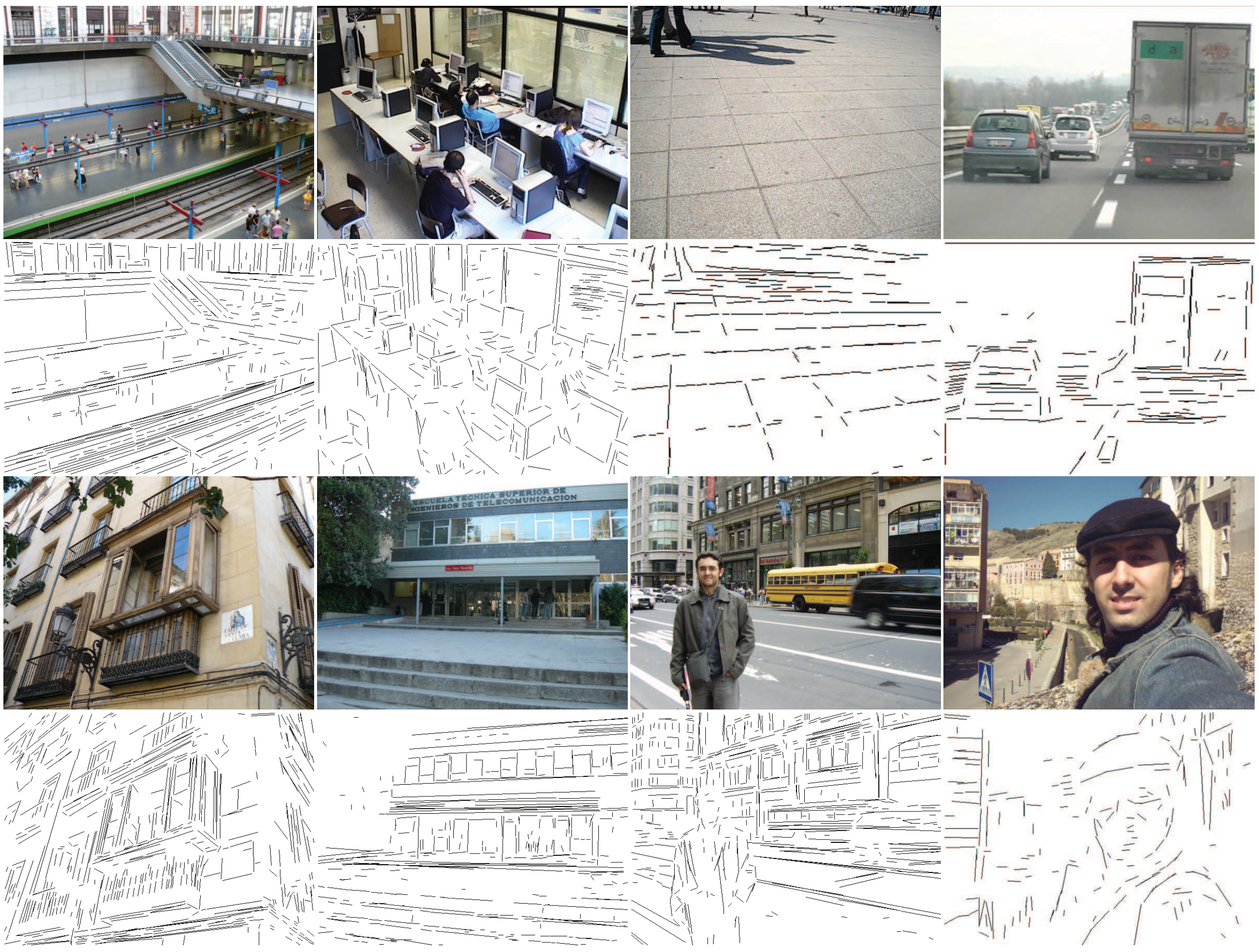


Fig. 8 Some results of the SSWMS applied on different real images.

$O(10N)$ 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 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.

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. Fig. 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 nega-

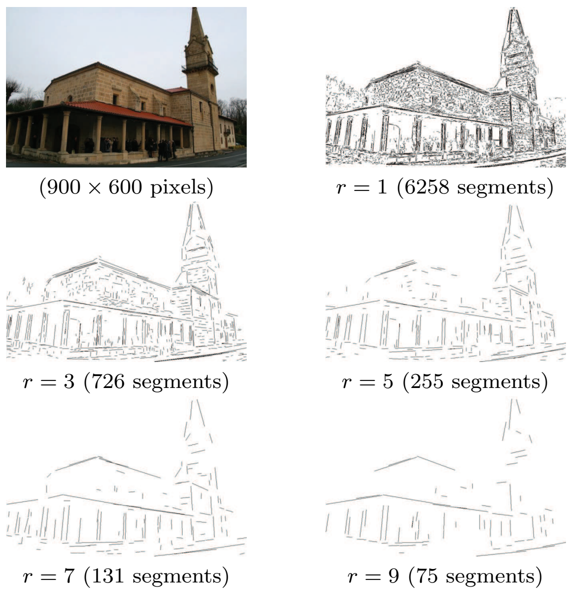


Fig. 9 Results of the SSWMS line segment detection using different values of the Mean Shift spatial component of the bandwidth, r.

tives or miss-detections. 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 leafs 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 1280×720, 1440×1080 or 1920×1080 require processing times of 430 ms (2.32 fps), 650 (1.5 fps), and 1000 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 1600×1200 pixels, would spend near 1000 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 bandwidth is an indicator of the resolution of the detector, i.e., the minimum distance between two detected line segments. For

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

	Requested number of line segments			
Image Size	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
1024 × 720	281	281	313	390
1366 × 768	391	391	422	563
1280 × 1024	485	484	531	688
1680 × 1050	641	656	687	922
1600 × 1200	703	703	735	1000
1900 × 1200	812	844	846	1172
3072 × 2304	2578	2578	2609	3781

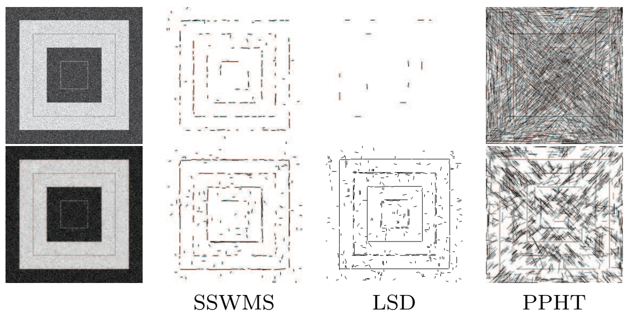


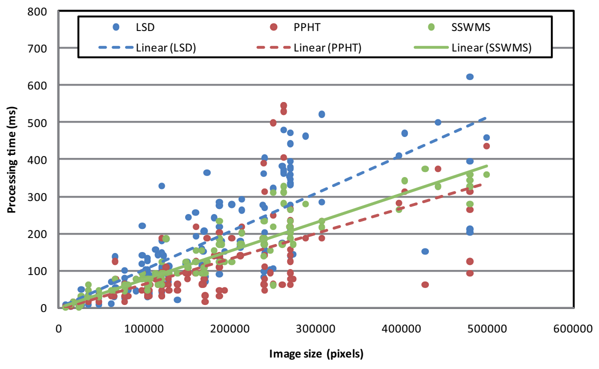
Fig. 10 Comparison of the performance of the compared methods against noise. The upper row shows the results obtained from the noisy image; and the second row shows the results for the smoothed version of the image.

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.

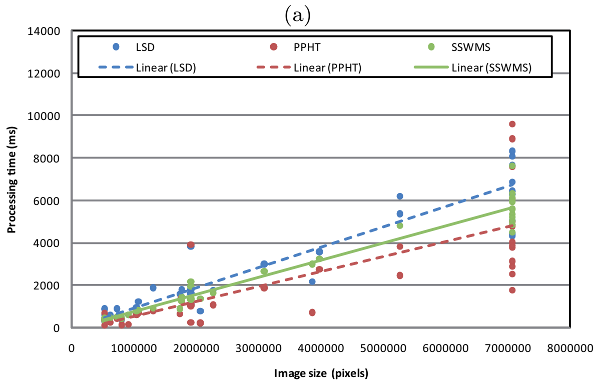
7.3 Comparison with other methods

We have selected two line segment detection algorithms to compare with our method: the LSD (Line Segment Detector [14]) and the PPHT (Progressive Probabilistic Hough Transform [12]) 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



(a)



(b)

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 3702 × 2304 pixels. The colored lines represents the linear tendency of the data along the image size axis. As shown, the faster method is the PPHT, while the SSWMS shows higher performance than LSD in both cases.

line segments in the image. In the second row, a gaussian filter is used to remove partially the noise, as suggested by [14] 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.

Fig. 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 large images. The PPHT is in 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

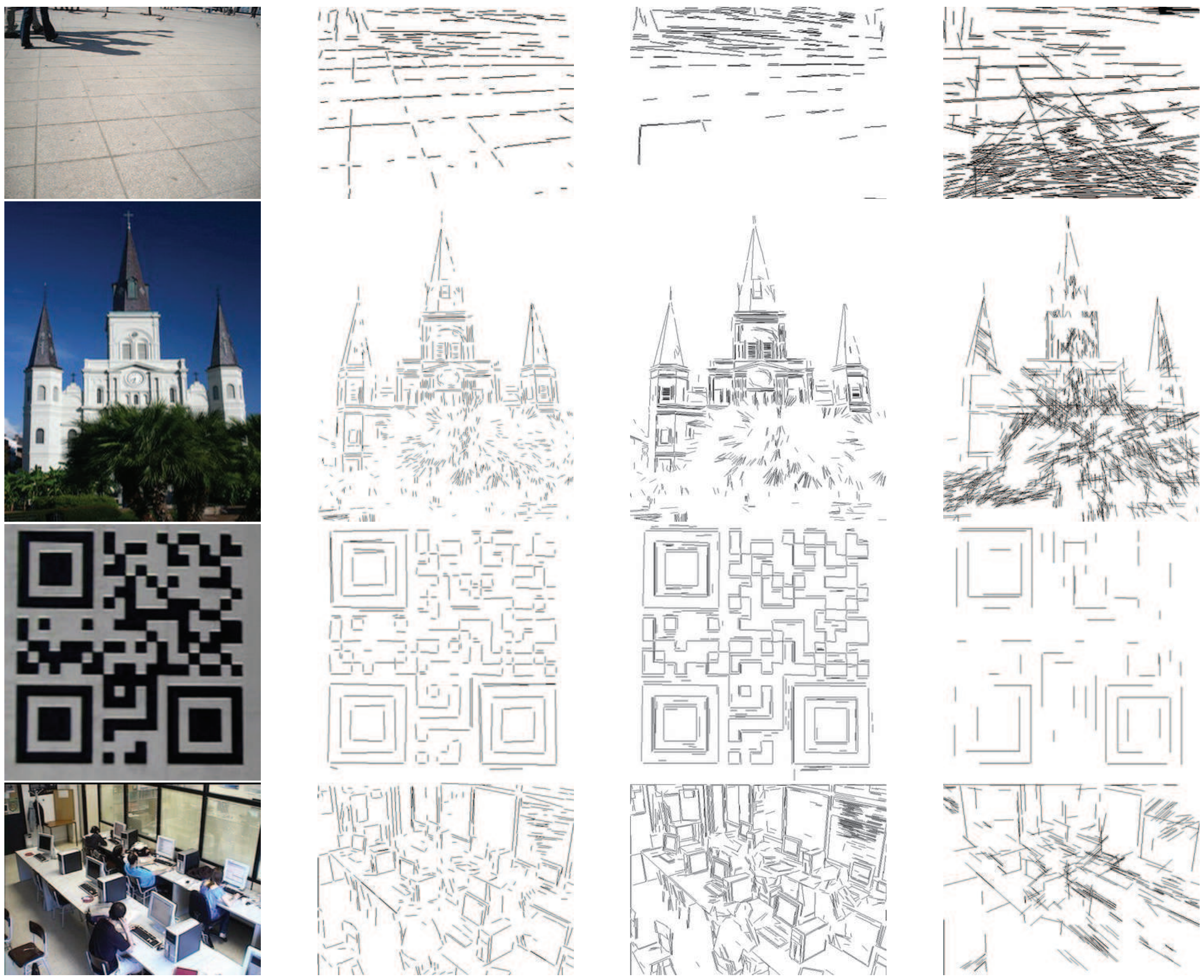


Fig. 12 Comparisons of the line segments obtained for different images applying different methods.

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 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 very reduced number of false detections and missdetections.

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 missdetections, and precision to the number of false alarms. These evaluation metrics can be computed as:

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

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

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: (i) variable size, (ii) noise, and (iii) 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

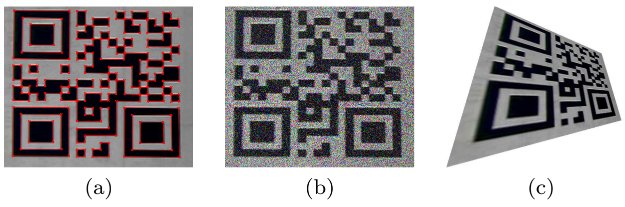


Fig. 13 Example image used to test recall and precision: (a) original image with ground truth line segments; (b) image with added gaussian noise; and (c) image with perspective distortion.

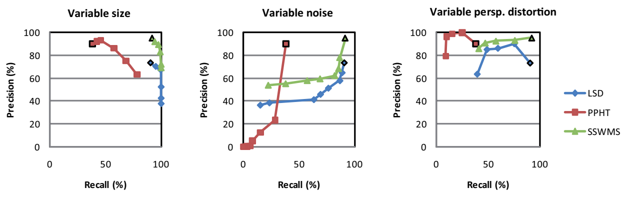


Fig. 14 Recall & Precision graphs for different situations: varying size, adding noise and perspective distortion for the three methods. The starting point is marked with a thick black edge, and is the same for the three graphs.

how flexible are the algorithms to provide 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. 13 (a), and for a number of modifications of it, illustrated in Fig. 13 (b) and (c), which follow the abovementioned 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 graphs[2]. 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. 13 (b). Finally, "variable perspective distortion" illustrates the results obtained by applying perspective transformations on the image, such as the one shown in Fig. 13 (c).

The results related to the modification of the size of the image indicates that the SSWMS renders very accurate results with very low missdetections 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.

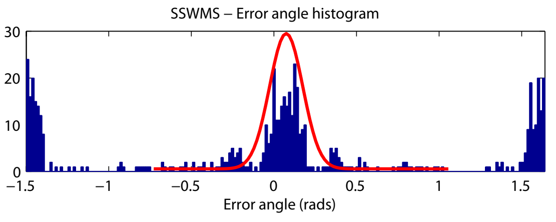
The results corresponding to the images with perspective distortion show that the recall values decrease as the number of line segments whose length is reduced grows. Nevertheless, the SSWMS algorithm still keeps a very good value of RP especially compared with the PPHT, which is not able to detect a large number of line segments due to the distortion.

8 Example application

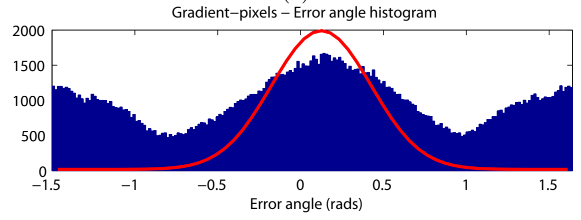
As commented in the introduction, line segments are useful image features for a number of applications in the computer vision field. As an example, we have successfully used the proposed SSWMS algorithm for the computation of vanishing points in sequences of images targeting plane rectification and camera autocalibration [27]. The accuracy of the SSWMS has been evaluated computing the orientation error of the obtained line segments with respect vanishing points.

For this purpose we have used the York Urban Data Base (YUDB) [11], 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. This error is related to the angle between the line segment and the line that joins the vanishing point and the mid-point of the line segment [27]. Fig. 15 (a) shows an example histogram of the error of the detected line segments with respect 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 correspond 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

[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 missdetections and thus its recall value is very low.



(a)



(b)

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

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. 15 (b). 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, 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,

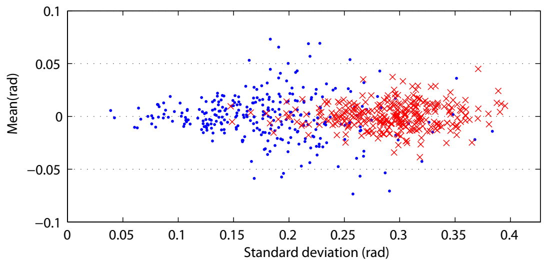


Fig. 16 Scatter plot of the mean and standard deviation of the normal fit of the inliers mode of the error histogram. The results of the normal fit of the line segments obtained with SSWMS is shown in dots. The normal fit corresponding to gradient-pixels is shown with crosses. Realize that there are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

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 need of tuning any parameter.

Acknowledgements 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. N. Aggarwal, and W. C. Karl. Line detection in images through regularized Hough transform. IEEE Transactions on Image Processing, vol. 15, no. 3, pp. 582-591, 2006.
2. A. Almansa and A. Desolneux and S. Vamech. Vanishing point detection without any a priori information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(4):502-507, 2003.
3. A. Bandera, J.M. Perez Lorenzo, J.P. Bandera, and F. Sandoval. Mean shift based clustering of Hough domain for fast line segment detection. Pattern Recognition Letters, 27(6):578-586, 2006.
4. C.M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, August 2006.
5. J. Bresenham. Algorithm for computer control of a digital plotter. IBM Systems Journal, 4(1):25-30, 1965.
6. J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting straight lines. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(4):425-455, 1986.
7. J. Canny. A comptational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679-698,1986.
8. D. Comaniciu, and P. Meer. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24:603-619, 2002.
9. B. Han, D. Comaniciu, Y. Zhu, and L.S. Davis. Sequential kernel density approximation and its application to real-time visual tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(7):1186-1197, 2008.

10. R. Dayhot. Statistical Hough Transform. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 8, pp. 1502-1509, 2009.
11. P. Denis, J.H. Elder, and F.J. Estrada. Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery. In IEEE Proc. European Conference on Computer Vision, vol. 2, pp. 197-210, 2008.
12. C. Galambos, J. Kittler, and J. Matas. Progressive probabilistic hough transform for lie detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1:1554, 1999.
13. W. Gilks, S. Richardson, and D. Spiegelhalter. Markov Chain Monte Carlo Methods in Practice. Chapman and Hall/CRC, 1996.
14. R. Grompone von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 99(1), 2009.
15. D.S. Guru, B.H. Shekar, and P. Nagabhushan. A simple and robust line detection algorithm based on small eigenvalue analysis. Pattern Recognition Letters, vol. 25(1), pp. 1-13, 2004.
16. M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer. Comparison of edge detectors: a methodology and initial study. Computer vision and image understanding, vol. 69, no. 1, pp. 38-54, 1998.
17. H. Kälviäinen, P. Hirvonen, L.Xu, and E. Oja. Comparisons of probabilistic and non-probabilistic Hough transforms. Proceedings of 3rd European Conference on Computer Vision, 351-360, 1994.
18. P. Khan, L. Kitchen, and E.M. Riseman. A fast line finder for vision-guided robot navigation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12(11), pp. 1098-1102, 1990.
19. N. Kiryati, Y. Eldar, and A.M. Bruckstein. A probabilistic hough transform. Pattern Recognition, 24(4):303-316, 1991.
20. J. Košecká, and W. Zhang. Video compass. European Conference on Computer Vision, LNCS 2350, 476-491, Springer Verlag, 2003.
21. D. Liebowitz, and A. Zisserman. Metric rectification for perspective images of planes. In IEEE Proc. Computer Vision and Pattern Recognition, vol. 0, pp. 482-488, 1998.
22. D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. Computer Graphics Forum, 18(3):39-50, 1999.
23. C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimation of vehicle velocity and traffic intensity using rectified images. IEEE International Conference on Image Processing, 777-780, 2008.
24. P.F.M. Nacken. A metric for line segments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:1312-1318, 1993.
25. R. Neal. Slice sampling. Annals of Statistics, 31:705-767, 2000.
26. R. Nevatia and K.R. Babu. Linear feature extraction and description. Computer Graphics and Image Processing, vol. 13, pp. 257-269, 1980.
27. M. Nieto, and L. Salgado. Real-time robust estimation of vanishing points through nonlinear optimization. In IST/SPIE Proc. International Conference on Real-Time Image and Video Processing, 2010.
28. R. Pflugfelder. Self-calibrating cameras in video surveillance. PhD thesis, Graz University of Technology, 2008.
29. E. Rosten, and T. Drummond. Machine learning for high-speed corner detection. European Conference on Computer Vision, 1:430-443, 2006.
30. D. Serby, E.K. Meier, and L.Van Gool. Probabilistic object tracking using multiple features. Proceedings of the 17th International Conference on Pattern Recognition, 2:184-187, 2004.
31. R.S. Stephens. Probabilistic approach to the hough transform. Image Vision Computing, 9(1):66-71, 1991.
32. D. Walsh, and A.E. Raftery. Accurate and efficient curve detection in images: the importance sampling Hough transform. Pattern Recognition, vol. 35, pp. 1421-1431, 2002.
33. H. Wang, Y. Cheng, T. Fang, J. Tyan, and N. Ahuja. Gradient adaptive image restoration and enhancement. In IEEE Proc. International Conference on Image Processing, pp. 2893-2896, 2006.
34. L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough Transform (RHT). Pattern Recognition Letters, vol. 11, no. 5, pp. 331-338, 1990.
35. S.Y.K. Yuen, T.S.L. Lam, and N.K.D. Leung. Connective Hough transform. Image and Vision Computing, vol. 11, pp. 295-301, 1993.