Connectivity-Enforcing Hough Transform for the Robust Extraction of Line Segments

Rui F. C. Guerreiro and Pedro M. Q. Aguiar, Senior Member, IEEE

Abstract

Global voting schemes based on the Hough transform (HT) have been widely used to robustly detect lines in images. However, since the votes do not take line connectivity into account, these methods do not deal well with cluttered images. In opposition, the so-called local methods enforce connectivity but lack robustness to deal with challenging situations that occur in many realistic scenarios, *e.g.*, when line segments cross or when long segments are corrupted. In this paper, we address the critical limitations of the HT as a line segment extractor by incorporating connectivity in the voting process. This is done by only accounting for the contributions of edge points lying in increasingly larger neighborhoods and whose position and directional content agree with potential line segments. As a result, our method, which we call STRAIGHT (Segment exTRAction by connectivity-enforcInG HT), extracts the longest to connected segments in each location of the image, thus also integrating into the HT voting process the usually separate step of individual segment extraction. The usage of the Hough space mapping and a corresponding hierarchical implementation make our approach computationally feasible. We present experiments that illustrate, with synthetic and real images, how STRAIGHT succeeds in extracting complete segments in several situations where current methods fail.

Index Terms

Hough transform, line segment detection, connectivity, connected segments, line pattern analysis, edge analysis.

EDICS Category: ARS-RBS

The authors are with the Institute for Systems and Robotics / Instituto Superior Técnico, Lisboa, Portugal. Contact author: P. Aguiar, Ph.: +351-21-8418283, Fax: +351-21-8418291, e-mail: aguiar@isr.ist.utl.pt.

I. INTRODUCTION

Line segments are fundamental low-level features for the analysis of many real-life images. In fact, most man-made objects are made of flat surfaces, originating images with edge maps composed by line segments. Thus, these segments provide important information about the geometric content of the imaged scene. This has been exploited, *e.g.*, for localizing vanishing points [1] or to match line segments across distinct views [2]. Since more elaborated shapes are often described in an economic way in terms of line segments, their extraction is often a first step in many other problems, *e.g.*, rectangle detection [3], the inference of shape from lines [4], map-to-image registration [5], 3D reconstruction [6], or even image compression [7].

In this paper, we propose a new method to extract line segments from images in an automatic way. Although this problem has been the focus of attention of several researchers in the past decades, current solutions do not cope with many challenging situations that arise in practice, as we detail in the sequel. For this reason, the robust detection of line segments remains an open frontier (see [8], [9] for examples of recent advances).

A. Overview of methods for line segment extraction

The Hough transform (HT) [10], [11] is the most popular method to detect lines in images. Basically, the HT extracts the lines that contain larger number of edge points. The key to an efficient implementation is the usage of the Hough space, a two-dimensional space parameterized in such a way that each point in this space represents a line in the image. Each edge point in the image is then mapped to the region of the Hough space that represents the pencil of all the image lines that go through that edge point. By processing all edge points, the votes for each location in the Hough space are accumulated (this space is also referred to as the accumulator array) and the locations with larger number of votes correspond to the parameterizations of the predominant lines in the image. Naturally, the success of the HT comes from its global nature, since all points in a line contribute to its detection.

A strong limitation of the HT comes from the fact that the voting scheme does not take into account that lines are alignments of *connected* points. In fact, since edge points voting for a particular line may be disconnected from each other, there is not guarantee that parameters receiving large numbers of votes correspond to long lines in the image (due to textures and/or noise, a peak in the accumulator array may even correspond to the parameterization of a "false" line, *i.e.*, a line that collects many separate points but is not at all perceived as a line in the image). This problem is particularly critical when using the HT to extract a line *segment*, *i.e.*, a rectilinear point alignment with length that can be much smaller than

the image dimensions. In first place, short segments originate small peaks in the Hough space, which are hard to identify [12]; besides, a non-trivial extra step to determine the segment start and end points is required, *e.g.*, the analysis of the shape of the spread of votes in the accumulator array [13].

Fig. 1 illustrates the difficulty that may arise when using the HT to detect short line segments. We contrast the HT of a synthetic image with long line segments (top left) with the one of a complex real image (bottom left). In the first case, since the number of lines is not very large and they are very long, the HT accumulator array exhibits the expected peaks (darker points visible in the top middle image), which are easily detected, and correspond to the correct lines (top right). In the second case, the large number of edge points in the real image, many of them forming very short segments or only corresponding to texture or noise, originates large numbers of votes for lines that do not correspond to connected segments. As a consequence, the accumulator array does not exhibit prominent peaks (see the smoothness of its grey-level representation in the bottom middle image) and its processing originates a poor result (bottom right) that contains spurious short segments and misses several others, including longer ones.



Fig. 1. Top: success of the HT when detecting long lines in a clutter-free image. Bottom: limitation of the HT when detecting short line segments in a complex image. In each case: on the left, the original image, on the middle, the HT accumulator arrays, and on the right, the detected line segments.

The limitations of the HT have been pointed out by several authors, *e.g.*, [12], [14], [15], [16] and many efforts have been made to alleviate its problems. For example, the method in [17] uses the edge direction to reduce the accumulation of spurious votes in the Hough space and [12] proposes a strategy

that is based on processing a line a time: after detecting the line corresponding to the largest peak in the accumulator array, the votes of the edge points that belong to this line are removed. Naturally, the success of this approach hinges on the correctness of the first lines detected. As a consequence, in many practical situations, when facing highly textured images and/or in the presence of noise, these approaches are unable to provide accurate results. In fact, this happens because these improvements do not tackle the fundamental limitation of the HT: the fact that it does not take into account the required connectivity of the edge points forming a line segment. Other authors addressed storage and computational issues of the HT by proposing a hierarchical scheme [18], multiple accumulator resolutions [19], or the usage of a random sampling of the edge map (probabilistic HT) [20].

In spite of these limitations, the global nature of the HT is attractive, since line parameters estimated from the complete data are naturally more accurate than what can be done locally. However, few papers have approached the problem of developing global methods for line segment extraction, *i.e.*, global methods to extract, simultaneously, the line parameters and its extremes. A fruitful example is the method in [21], which searches among all possible line segment candidates, using a Helmholtz principle for validating. Naturally, the good results come at a high computational cost. Other approaches use the widely known random sample consensus (RANSAC) [22]: basically, two edge points from the edge map, randomly sampled, define a candidate line; then, the consensus of the line is evaluated by counting the number of other edge points that fit that line segment, given an error tolerance; for segments with a high consensus, the parameters are refined by using an iterative expectation-maximization (EM) method [23]. Since the success of RANSAC hinges on the usage of a very large number of samples, these approaches result computationally too complex for many realistic applications.

For the reasons above, the majority of methods for line segment extraction rely on local decisions, rather than on global ones, see [24], [25], [14], [26], [16] for examples. These local methods outperform the HT by taking (local) connectivity into account, and result computationally simple, but lack robustness to deal with challenging situations, *e.g.*, when line segments cross. Furthermore, their local nature make long line segments particularly difficult to extract in many realistic scenarios, because, due to noise and clutter, these segments are interrupted. The majority of local methods use three steps: first, obtaining a region of connected edge points; then, roughly estimating the line segment direction; and finally, refining and extending the segment by including new edge points that approximately fit the line. The first step consists of chaining edge points [27]. Methods such as the one in [28] even skip the subsequent line fitting and refinement steps by chaining connected edge points into curves and then cutting them into line segments, using a straightness criterion. Texture, low-contrast regions, crossing segments, and noise

make difficult the extraction of large connected regions belonging to a single segment. The second step consists of fitting a line to the chain of edge points using, e.g., total least-squares (TLS) [26]. Naturally, the reliability of the regression depends on the length of the underlying point chain. Some methods bypass the chaining of edge points: [29] uses the so-called local HT (LHT) [30], roughly estimating the segment direction from the peaks of local orientation histograms, computed at each edge point; [14], [31] directly fit a line to all edge points inside a sliding window, which only provides reasonable estimates for simple scenes, with very small clutter. The final step usually involves alternating between two stages until convergence [26]: inclusion of new edge points that are close to the candidate line, according to a distance measure; and re-estimation of the line segment parameters from the new set of edge points. As it is typical with this type of methods, a poor initial model for the line segment model may compromise the final result. Furthermore, the process may terminate too early when attempting to extract a long line segment, due to the common cluttered nature of the edge maps of real images. Two popular local methods for line segment detection are [25] and the LSD (Line Segment Detector) of [16]. The method in [25] coarsely quantizes the local orientation angles, chains adjacent pixels with identical orientation labels, and fits a line segment to the grouped pixels. LSD [16] extends this idea by using continuous angles and eliminates false line segment detections by using the Helmholtz principle of [21].

B. Proposed approach

The key ingredient of our method is the incorporation of connectivity into the HT voting process. This is done by imposing that edge points only vote for lines according to which they are spatially connected to other points. As a consequence, the vast majority of spurious votes are eliminated and peaks in the accumulator array become prominent and truly correspondent to line segments of maximum length. Simultaneously, our method, which we call STRAIGHT (Segment exTRAction by connectivity-enforcInG Hough Transform), integrates into the voting process the usually separate step of determining the extremes.

STRAIGHT starts by computing the prominent directions at each edge point, which will guide the search for the orientations of line segments. An image line segment is characterized by a rectilinear alignment of dark-to-light (or opposite) transitions. Our prominent direction detector computes, for each edge point, the set of directions according to which there is a predominance of those intensity transitions. This is accomplished by extending the LHT [30] to only take into account directionally coherent edge points: in a first step, signed directional edge maps are computed; then, orientation histograms are built at each edge point, by considering the neighboring edge points whose relative positions agree with the angle

6

of the directional edge map. The histogram accumulates the signed values of the intensity transitions, thus the prominent directions at each edge point are detected by finding large magnitude entries in the corresponding histogram.

After computing the local prominent directions, STRAIGHT extracts line segments using the knowledge that the edge points forming each of them must be connected. This is done by computing new LHT-like maps (which we will call *length maps*) for each edge point, this time taking into account all other edge points whose position and directional content agree with potential line segments. Position matters because only points that respect connectivity are considered; directional content matters because only edge points with prominent direction that agrees with the candidate segment are considered. In practice, for each prominent direction of each edge point, STRAIGHT progressively considers edge points further away until the connectivity criterion is violated. After exploring all candidate directions, the ones that collected more distant edge points correspond to the orientations of the longest connected line segments going through the starting point. Note that allowing a set of prominent directions, rather than a single one, enables dealing with crossing segments.

Our implementation of STRAIGHT incorporates the explicit mapping of uncertainty balls around the edge points into the Hough space, increasing robustness and accuracy, and uses a hierarchical coarse-to-fine strategy to explore candidate directions, leading to a computationally tractable algorithm. We present illustrative results of experiments that use synthetic and real images to compare STRAIGHT with the HT [11] and the state-of-the-art local method LSD [16]. A preliminary version of this work is in [32].

C. Paper organization

The organization of the remaining of the paper is as follows. Section II describes the computation of local prominent directions. Section III details the extraction of line segments by enforcing connectivity. The hierarchical implementation is described in Section IV. Experimental results are reported in Section V and Section VI concludes the paper.

II. COMPUTING LOCAL PROMINENT DIRECTIONS

In terms of image intensities, the existence of a line segment corresponds to the presence of a rectilinear alignment of dark-to-light (or opposite) transitions. Despite the multiple sources of inaccuracies in edge detection, such as noise and clutter, there should be a predominance of either positive or negative intensity variations (corresponding to light-to-dark and dark-to-light transitions, respectively) along the

7

line segment. This predominance is what we exploit to compute the prominent directions at each edge point, *i.e.*, the set of possible orientations of line segments going trough that point.

We capture the local directional content of an image I by computing its derivatives, through the convolution with four oriented kernels,

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \boldsymbol{I} = \boldsymbol{I} \ast \boldsymbol{K}_{\boldsymbol{\theta}}, \quad \boldsymbol{\theta} \in \{0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}\}.$$
(1)

Although kernels with a large support would smooth the noise, we use simple standard central difference kernels, since they enable more precise edge localization and angular responses, by minimizing the influence of surrounding pixels. Thus, we set

$$\boldsymbol{K}_{0} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{K}_{45} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \ \boldsymbol{K}_{90} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \ \boldsymbol{K}_{135} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In our case, robustness to noise when extracting line segments comes from the requirement of point connectivity, as detailed in the following section.

The directional edge maps E_{θ} , $\theta \in \{0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}\}$, are obtained by thresholding the derivatives and retaining their sign, *i.e.*,

$$\boldsymbol{E}_{\theta}(x,y) = \begin{cases} 1 & \text{if } \boldsymbol{\nabla}_{\theta} \boldsymbol{I}(x,y) \geq T \\ -1 & \text{if } \boldsymbol{\nabla}_{\theta} \boldsymbol{I}(x,y) \leq -T \\ 0 & \text{otherwise} \,. \end{cases}$$

We call (x, y) an edge point when $|E_{\theta}(x, y)| = 1$ for at least one value of θ .

We extend the LHT [30] to take into account the edge directional content, *i.e.*, our method builds local orientation histograms by using the neighboring edge points whose direction is coherent with its position. To clarify, when building the histogram for the edge point (x_0, y_0) , we first compute the directions of the segments passing through (x_0, y_0) and each neighboring edge point (x, y),

$$\theta_{(x_0,y_0)}(x,y) = \arctan\left(\frac{y-y_0}{x-x_0}\right) \in [0^\circ, 180^\circ).$$

Then, for each neighbor (x, y), the histogram count uses only the directional edge map E_{θ} with the value of θ that is closer to direction perpendicular to $\theta_{(x_0,y_0)}(x, y)$, *i.e.*,

$$E_{90}(x,y) \quad \text{if} \quad \theta_{(x_0,y_0)}(x,y) \in [0,22.5] \cup (155.5,180) ,$$

$$E_{135}(x,y) \quad \text{if} \quad \theta_{(x_0,y_0)}(x,y) \in (22.5,67.5] ,$$

$$E_0(x,y) \quad \text{if} \quad \theta_{(x_0,y_0)}(x,y) \in (67.5,112.5] ,$$

$$E_{45}(x,y) \quad \text{if} \quad \theta_{(x_0,y_0)}(x,y) \in (112.5,155.5] ,$$
(2)

which just corresponds to the representation in Fig. 2. For example, if $(x_0, y_0) = (0, 0)$ and (x, y) = (0, 3), we have $\theta_{(0,0)}(0,3) = 90^\circ$, a vertical segment, and the directional edge point that contributes to the histogram is $E_0(0,3)$, since K_0 is the kernel that best responds to the horizontal transitions that define vertical edges.



Fig. 2. Selection of directional edge map E_{θ} in terms of the relative position between edge points.

As usual in the LHT, the number B of histogram bins is fixed (B = 32 is typical) and each edge point contributes to the two bins whose centers approximate the angle $\theta_{(x_0,y_0)}(x,y)$ by excess and default (the contributions are weighted according to the distance to the bin centers). The signs in the directional edge maps are taken into account through positive or negative contributions to the histogram bins. This way, we filter out conflicting contributions that may occur due to noise, textures and image clutter. We use a relatively large neighborhood for the local histograms (a 7-pixel radius circular window) to minimize the influence of the spurious points in the edge maps. The prominent directions at each edge point are found by thresholding the magnitude of the corresponding local histogram (we use the threshold of 50% of the number of edge points inside the circular window for each direction, thus a direction is considered prominent when the majority of the edge points in that direction have the same intensity transition sign).

To represent the range of possible directions going through each edge point (x_0, y_0) , we store the set of prominent bin centers and the corresponding image gradients, *i.e.*,

$$\boldsymbol{\Theta}(x_0, y_0) = \{ (\theta_1, \boldsymbol{\nabla}_{\theta_1} \boldsymbol{I}(x_0, y_0)), (\theta_2, \boldsymbol{\nabla}_{\theta_2} \boldsymbol{I}(x_0, y_0)), \dots, (\theta_N, \boldsymbol{\nabla}_{\theta_N} \boldsymbol{I}(x_0, y_0)) \}$$

where $0 \le N \le B$ is the number of histogram bins whose count was above the threshold. If $N \ge 1$, for $1 \le n \le N$, we denote by θ_n the central angle of the bin n and by $\nabla_{\theta_n} I(x_0, y_0)$ the image gradient (1), with orientation θ that best matches θ_n , according to (2). To account for the histogram discretization, *i.e.*, the nonzero width of the bins, we consider in the sequel as possible directions of line segments all the orientations $\theta \in [\theta_n - \Delta_{\theta}, \theta_n + \Delta_{\theta}]$, with $\Delta_{\theta} = 180/B/2 = 90/B$.

III. EXTRACTING CONNECTED LINE SEGMENTS

We now describe the core of STRAIGHT, *i.e.*, the way we incorporate connectivity into the line segment extraction. We start by making explicit the parameter search problem that underlies the extraction of each of the segments, introducing the *length map*, which plays a similar role of the HT accumulator array. Then, we describe how edge points are sequentially mapped into the length map by taking into account both the edge point connectivity and the uncertainty due to discretization. Finally, we describe the procedure to extract line segments from the length map.

A. Line segment extraction as a parameter search problem – the length map

Let $p_0 = (x_0, y_0)$ be an edge point and $\Theta(p_0)$ the set of prominent directions of possible line segments passing through p_0 , as introduced in the previous section. To accurately detect a line segment in the image it is necessary to estimate the sub-pixel location of the line, which will be parameterized by its position and orientation, in a similar way to the HT. The line position is specified in terms of its distance δ_p to the edge point p_0 . The line orientation is represented by δ_{θ} , which represents the deviation with respect to the prominent direction angle θ_n in $\Theta(p_0)$. Thus, as illustrated in Fig. 3, any point p = (x, y) belonging to the line $(\delta_p, \delta_\theta)$ obeys

$$\langle \boldsymbol{p}, \boldsymbol{v}_{\theta_n + \delta_\theta}^{\perp} \rangle = \langle \boldsymbol{p}_0, \boldsymbol{v}_{\theta_n + \delta_\theta}^{\perp} \rangle + \delta_p , \qquad (3)$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product and $v_{\theta}^{\perp} = (\sin(\theta), -\cos(\theta))$ is a unit vector with directional orthogonal to θ . The candidate line segment is allowed to deviate at most a predefined Δ_p from p_0 , *i.e.*, $\delta_p \in [-\Delta_p, \Delta_p]$, (in our experiments, we use $\Delta_p = 1$) and Δ_{θ} from θ_n , *i.e.*, $\delta_{\theta} \in [-\Delta_{\theta}, \Delta_{\theta}]$ (Δ_{θ} was defined in the previous section).



Fig. 3. Edge point p_0 , prominent direction θ_n , line segment specified by parameters $(\delta_p, \delta_\theta)$, and range limits Δ_p and Δ_{θ} .

When the estimate $(\delta_p, \delta_\theta)$, coincides with the true value of the parameters defining a line segment in the image, there are several other edge points along the line $(\delta_p, \delta_\theta)$ for which the orientation $\theta = \theta_n + \delta_\theta$ is also prominent (with matching signs of the image gradient), according to the information collected in $\{\Theta(\cdot)\}$. We call a *candidate match* to each of these edge points. Besides, due to the connectivity of the edge points forming a line, the gap between candidate matches must be smaller than a predefined *maximum distance threshold d*. Naturally, the closer the estimated line is to the actual one, more distant edge points of the true line segment are captured. This is the key point of our approach, which formalizes the extraction of a line segment as the search for the parameters $(\delta_p, \delta_\theta)$ that *maximize the length* of the segment that can be extracted in the neighborhood of p_0 .

To extract the maximum length segments that pass close to each edge point, we borrow inspiration in the LHT, where local accumulator arrays are used in contrast to the single accumulator array of the HT, which can not discriminate between distinct segments falling in the same line. In our case, we define local 2D *length maps* $\boldsymbol{L} : [-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta] \mapsto \mathbb{N}_0$. For each edge point, $\boldsymbol{L}(\delta_p, \delta_\theta)$ will contain the integer length of the line segment $(\delta_p, \delta_\theta)$. This map can be regarded as an extension of the accumulator array of a LHT in order to take line segment connectivity into account. In this scenario, the extraction of a line segment passing close to \boldsymbol{p}_0 consists of filling $\boldsymbol{L}(\cdot, \cdot)$, obtaining the parameters $(\delta_p, \delta_\theta)$ for which $\boldsymbol{L}(\delta_p, \delta_\theta)$ is maximum, and computing its start and end points.

To fill $L(\cdot, \cdot)$ through the direct implementation of an exhaustive scanning of the space $[-\Delta_p, \Delta_p] \times [-\Delta_{\theta}, \Delta_{\theta}]$ would be computationally unbearable. In fact, the discretization of this space must be very fine to yield accurate results and, for each location, many edge points have to be processed (possibly, hundreds or thousands). Besides, this approach would use redundant computations, since each edge point would be visited several times because it may belong to several candidate line segments. For this reason, as usually done to fill the HT accumulator array, we also adopt a pixel-centered approach, where the edge points are used to fill the length map $L(\cdot, \cdot)$ in an efficient way.

B. Incorporating uncertainty – the update region

We now show how each individual edge point is processed in our pixel-centered approach. Due to the pixel grid discretization, we model each edge point by an uncertainty ball, rather than a pointwise feature. We use the uncertainty ball radius R = 1, the maximum expected error in the location of each pixel. Because the uncertainty ball has a non-infinitesimal area, there is a set of parameters $\{(\delta_p, \delta_\theta)\}$, whose corresponding line segments cross it. This is illustrated in the left side of Fig. 4, where the lines formed by all orientations between θ_a and θ_b cross the uncertainty ball centered at (x, y), for position deviation $\delta_p = -\alpha$ from p_0 . We call *update region* of the length map domain to the set of positions and orientations $\{(\delta_p, \delta_\theta)\}$, whose corresponding line segments cross the uncertainty ball centered at each edge pixel. The update region for the pixel p = (x, y), in the length map of p_0 , is shown on the right side of Fig. 4.



Fig. 4. The uncertainty ball of an edge pixel (left) and the corresponding update region in the length map domain (right).

To find the analytic expressions for the bounds of update region, we use the line equation (3), now seen as a condition for line segments rather than points. When computing the length map for the edge point p_0 , we see from (3) that any segment $(\delta_p, \delta_\theta)$ that crosses the uncertainty ball of pixel p = (x, y)must verify

$$\langle \boldsymbol{p}, \boldsymbol{v}_{\theta_n + \delta_\theta}^{\perp} \rangle = \langle \boldsymbol{p}_0, \boldsymbol{v}_{\theta_n + \delta_\theta}^{\perp} \rangle + (\delta_p - r) , \qquad (4)$$

where $-R \leq r \leq R$ is the distance between the center of the ball and the segment, along vector $v_{\theta_n+\delta_\theta}^{\perp}$ (depicted in Fig. 4). From (4), the line segment position δ_p is easily expressed in terms of its orientation δ_{θ} and r:

$$\delta_p = \langle \boldsymbol{p} - \boldsymbol{p}_0, \boldsymbol{v}_{\theta_n + \delta_\theta}^{\perp} \rangle + r \,. \tag{5}$$

The update region, which we denote by U, is thus the collection of intervals specified by all possible orientations δ_{θ} and corresponding positions δ_p given by (5), with $|r| \leq R$:

$$\boldsymbol{U} = \left\{ \left(\delta_p, \delta_\theta\right) : \delta_\theta \in \left[-\Delta_\theta, \Delta_\theta\right], \delta_p \in \left[\delta_p^-(\delta_\theta), \delta_p^+(\delta_\theta)\right] \cap \left[-\Delta_p, \Delta_p\right] \right\},\tag{6}$$

where the limits $\delta_p^-(\delta_\theta)$ and $\delta_p^+(\delta_\theta)$ are made explicit from (5) by expanding $v_{\theta_n+\delta_\theta}^{\perp}$:

$$\delta_p^-(\delta_\theta) = (x - x_0)\sin\left(\theta_n + \delta_\theta\right) - (y - y_0)\cos\left(\theta_n + \delta_\theta\right) - R\,,\tag{7}$$

September 19, 2011

DRAFT

$$\delta_p^+(\delta_\theta) = (x - x_0)\sin\left(\theta_n + \delta_\theta\right) - (y - y_0)\cos\left(\theta_n + \delta_\theta\right) + R.$$
(8)

If the update region of a given pixel is non-empty, we say that the pixel is within the *search range*. In the illustration of Fig. 4, the search range for p_0 is the one limited by the lines labelled with $\theta_n - \Delta_{\theta}$ and $\theta_n + \Delta_{\theta}$.

As geometrically evident, the range of angles of lines that cross an uncertainty ball decreases as the distance between p_0 and p increases (an approximate expression for this range is $2 \arcsin (R/||p - p_0||)$, obtained by noting that $\overline{p_0, p}$ can be approximated by the hypotenuse of a right-angled triangle of which R is the small cathetus). As a consequence, to enable the extraction of long line segments, *i.e.*, containing edge points p far from p_0 , the length map must be densely discretized to sample all relevant values of δ_{θ} . We propose an efficient way to deal with this need through the hierarchical coarse-to-fine procedure described in the Section IV.

We observe that expressions (7) and (8) are similar to the parameterizing equation of the HT [11], $\rho = x \cos(\theta) + y \sin(\theta)$, with $\mathbf{p}_0 = (x_0, y_0)$ as the origin of the coordinate system and θ shifted by 90°. Thus, the boundaries of the update region resemble the sinusoidal shape of the bundles of votes of each edge point in the HT accumulator array (see Fig. 4 where, in fact, only a segment of that shape is seen, due to the length map limits). In what respects to the resolution of the accumulator array, when using the HT, the contradictory requirements of accuracy (high resolution) and coping with discretization error (low resolution, so that votes of the same line fall within the same bin) makes difficult, if not impossible, to achieve a good compromise. Strategies that uniformly blur the accumulation array (*e.g.*, by using multiple resolutions [18], [19], or kernels of various sizes [8]) do not change the scenario, since they still neglect the distinct influence of the discretization error of edge points located at different positions. In opposition, in our case, the resolution of the length map can be chosen arbitrarily large, since we model the actual discretization error of each individual edge point by using the correspondent (position-dependent) update region, as descibed above.

C. Sequential mapping of edge points to the length map

After describing how each pixel maps to a corresponding update region in the length map, we now show how to fill this map in a sequential way by processing all image edge points. As before, let us consider the case the of the edge point p_0 , with prominent direction θ_n . When filling the corresponding length map L, we consider the image divided in two half-planes by the line orthogonal to θ_n passing through p_0 . In each half-plane, starting from p_0 we circularly scan the image, with progressively larger radius, mapping to the corresponding half-plane length map the candidate matches that fall within the search range and do not violate the connectivity requirement.

Due to the graceful adaptation to the discrete pixel grid, we use the so-called Manhattan distance to define the *equidistant curves* as the set of pixels p located at fixed distance e from p_0 (*i.e.*, such that $||p - p_0||_{\infty} = e$). Fig. 5 illustrates the scenario, with the central edge point p_0 , the equidistant curves, labeled by the distance values, the search range and the half-planes.



Fig. 5. Central edge point p_0 , search range, and equidistant curves, each labeled by its integer distance to p_0 .

For each half-plane, we scan each equidistant curve, starting with the one closer to p_0 (*i.e.*, the curve with label e = 1 in Fig. 5), looking for candidate matches. Fig. 6 illustrates the scanning pattern for each equidistant curve. It starts in the center of the search range, *i.e.*, the pixel labeled with 0 in Fig. 6, and processes each pixel within the curve until reaching the limit of the search range (*i.e.*, the pixels labeled with positive values in Fig. 6, up to label 4, shown in red). Then, the pixels in the other direction are scanned, until the complete equidistant curve within the search range was processed (*i.e.*, the pixels labeled with negative values in Fig. 6, down to label -6, shown in red). Then, the following equidistant curve is processed.

To account for the usage of the Manhattan distance, the discrete pixel [p] at the center of the search range for the equidistant curve e is given by

$$[p] = \operatorname{round} \left(\boldsymbol{p}_0 \pm e rac{ \boldsymbol{v}_{ heta_n} }{\| \boldsymbol{v}_{ heta_n} \|_\infty}
ight) \,,$$

where $v_{\theta} = (\cos(\theta), \sin(\theta))$ is a unit vector with angle θ , and the signal \pm depends on the half-plane being considered.



Fig. 6. Illustration of the scanning pattern for a single equidistant curve in one of the half planes.

When a candidate match is found in the equidistant curve e, its update region is computed, according to (6), (7), and (8), and the corresponding entries of the length map are updated. This updating consists simply in setting those entries to the value of the equidistant curve number, e. This indicates that there are valid line segments of (at least) size e with the parameters corresponding to those entries. To capture the connected nature of the line segments, we first prune the update region, eliminating the locations where the difference between the current value of the length map, L, and the equidistance value e is larger than the maximum distance threshold d, *i.e.*,

$$\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \{ [e - \boldsymbol{L}(\delta_p, \delta_\theta)] > d, \delta_p \in [-\Delta_p, \Delta_p], \delta_\theta \in [-\Delta_\theta, \Delta_\theta] \} .$$
(9)

Then, we update the length map according to

$$\boldsymbol{L} \leftarrow e\boldsymbol{U} + \boldsymbol{L} \odot \boldsymbol{U}$$
,

where U is seen as a binary mask and \odot denotes the Hadamard, or elementwise, product. When the distance between e and all the values in the length map, $L(\cdot, \cdot)$, is larger than d, *i.e.*, when $U = \emptyset$, there are not updatable entries in the length map and the scanning stops for the corresponding half-plane. In the vast majority of our experiments, we used d = 2 pixels.

Alg. 1 synthesizes the procedure just described to compute each half-plane length map. The final length map for each edge point p_0 and prominent direction θ_n is obtained by adding the two half-plane length maps.

Algorithm 1 Filling one half-plane length map L for edge point p_0 and prominent direction θ_n .

1: input: $p_0 = (x_0, y_0), \theta_n$, prominent directions $\{\Theta(\cdot)\}$, maximum distance d, uncertainty radius R 2: $L(\cdot, \cdot) = 0, e = 1$ 3: repeat for $[\mathbf{p}] = (x, y)$ in the equidistant curve e (as illustrated in Fig. 6) do 4: if [p] is a *candidate match* (according to $\Theta(p)$) then 5: $U = \emptyset$ 6: for $\delta_{\theta} \in [-\Delta_{\theta}, \Delta_{\theta}]$ do 7: $\delta_n^- = (x - x_0) \sin(\theta_n + \delta_\theta) - (y - y_0) \cos(\theta_n + \delta_\theta) - R$ 8: $\delta_p^+ = (x - x_0) \sin \left(\theta_n + \delta_\theta\right) - (y - y_0) \cos \left(\theta_n + \delta_\theta\right) + R$ 9: $\boldsymbol{U} \leftarrow \boldsymbol{U} \cup \left\{ (\delta_p, \delta_\theta) : \delta_p \in \left[\delta_p^-, \delta_p^+ \right] \cap \left[-\Delta_p, \Delta_p \right] \right\}$ 10: end for 11: $U \leftarrow U \setminus \{e - L(\cdot, \cdot) > d\}$ (requirement of line segment connectivity) 12: $L \leftarrow eU + L \odot \overline{U}$ 13: end if 14: 15: end for 16: $e \leftarrow e + 1$ 17: **until** $e - \max\{L(\cdot, \cdot)\} > d$ 18: **output:** *L*

D. Extracting line segments

As when detecting lines from the peaks of the HT accumulator array, we detect line segments passing through p_0 with an orientation close to the prominent direction θ_n by simply collecting position-orientation pairs lying in the range $(\delta_p, \delta_\theta) \in [-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ that correspond to peaks in the corresponding length map $L(\cdot, \cdot)$.

Whenever a line segment is detected, with position-orientation parameters $(\delta_p, \delta_\theta)$, we also obtain in a straightforward way the coordinates of its extremes:

$$[p_{\pm}] = \operatorname{round} \left(\boldsymbol{p}_0 + E_{\pm} \frac{\boldsymbol{v}_{\theta_n + \delta_{\theta}}}{\|\boldsymbol{v}_{\theta_n + \delta_{\theta}}\|_{\infty}} + \delta_p \boldsymbol{v}_{\theta_n + \delta_{\theta}}^{\perp} \right) , \qquad (10)$$

where the subscript \pm differentiates both extremes and E_{\pm} denotes the maximum values of the length map of the corresponding half-planes.

16

To prevent multiple detections of a single line segment, each time a segment is detected for a central point p_0 and prominent direction θ_n , we remove that prominent direction from all candidate matches p in the line segment. Multiple crossing segments are naturally extracted by collecting position-orientation pairs in all prominent directions $\{\theta_n, 1 \le n \le N\}$. To enable the detection of crossing segments with very close direction angles, a fine discretization of the angle histogram is required. Alternatively, we can use variable bin sizes for each prominent direction, in which case only (a small length interval around) the angle of an extracted line would be removed from all candidate matches p in the line segment. In the latter scenario, the corresponding length map may contain more than one peak, thus each one is dealt with independently.

A final remark regards avoiding that a few edge points of lines with position-orientation parameters outside the range $[-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ vote for spurious lines inside that range. If fact, this would happen whenever the uncertainty balls of those edge points intersect that range, as illustrated in Fig. 7. We explicitly detect these cases and ignore them by using an orientation limit slightly larger than Δ_θ (say, an increase of 2°) and only consider as detected segments those with estimated orientation within the original limits, *i.e.*, $\delta_\theta \in [-\Delta_\theta, \Delta_\theta]$.



Fig. 7. Edge points in a line outside the position-orientation range $[-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ and a spurious line segment, shown in green, that could be erroneously detected inside that range.

Alg. 2 synthesizes the procedure to extract line segments, where the usage of *non-maxima suppression* in line 3 is not detailed, since it is similar to the standard procedure for extracting peaks from the accumulator array of the HT. The only difference is that, since STRAIGHT has detected all the candidate matches for each line segment, the parameters (δ_p , δ_θ) are more accurately estimated by fitting a line to the coordinates of the candidate matches with weights proportional to the magnitude of the image gradients. Naturally, other fitting criteria can easily be adopted in STRAIGHT. Algorithm 2 Extracting line segments passing through p_0 with orientation close to θ_n .

- 1: input: p_0 , θ_n , half-plane length maps $L_+(\cdot, \cdot)$ and $L_-(\cdot, \cdot)$, prominent directions $\{\Theta(\cdot)\}$, angle range limit Δ_{θ} , uncertainty ball radius R
- 2: repeat
- 3: $(\delta_p, \delta_\theta) = \arg \max [L_+(\cdot, \cdot) + L_-(\cdot, \cdot)]$ (find and remove peak using *non-maxima suppression*)
- 4: **if** $|\delta_{\theta}| \leq \Delta_{\theta}$ then

5:
$$[\boldsymbol{p}_{+}] = \operatorname{round} \left(\boldsymbol{p}_{0} + \boldsymbol{L}_{+}(\delta_{p}, \delta_{\theta}) \boldsymbol{v}_{\theta_{n}+\delta_{\theta}} / \| \boldsymbol{v}_{\theta_{n}+\delta_{\theta}} \|_{\infty} + \delta_{p} \boldsymbol{v}_{\theta_{n}+\delta_{\theta}}^{\perp} \right)$$

 $6: \qquad [\boldsymbol{p}_{-}] = \operatorname{round} \left(\boldsymbol{p}_{0} + \boldsymbol{L}_{-}(\delta_{p}, \delta_{\theta}) \boldsymbol{v}_{\theta_{n}+\delta_{\theta}} / \| \boldsymbol{v}_{\theta_{n}+\delta_{\theta}} \|_{\infty} + \delta_{p} \boldsymbol{v}_{\theta_{n}+\delta_{\theta}}^{\perp} \right)$

- 7: for the *candidate matches* p whose distance to $\overline{[p_-][p_+]}$ is smaller than (or equal to) R, remove from $\Theta(p)$ the entry corresponding to θ_n
- 8: **end if**
- 9: until there are not prominent peaks in $[L_{+}(\cdot, \cdot) + L_{-}(\cdot, \cdot)]$
- 10: **output:** extremes of the extracted line segments, $\{[p_-], [p_+]\}$, and updated $\{\Theta(\cdot)\}$

IV. HIERARCHICAL IMPLEMENTATION

Although the computational complexity of the pixel-centered approach described in the previous section is much smaller than an intensive approach, there are still some issues that need addressing. Because the discretization of $L(\cdot, \cdot)$ must be fine, every time a new candidate match is found, a very large amount of positions in the length map need to be updated, which is a time-consuming operation. Furthermore, the number of pixels in the equidistant curves that fall within the search range and need to be scanned increases considerably with the size of the detected segment. Simultaneously, as the scanning of edge pixels proceeds and the corresponding updates are incorporated in the length map, the region of the map that remains updatable progressively becomes smaller. This occurs because more distant pixels correspond to smaller angle ranges, as explained in the previous section, and fewer (δ_p , δ_θ) positions still correspond to quasi-connected line segments. Since this narrowing of the updatable area was not taken into account in the previous section, most pixel checks are unnecessary and further computational cost optimizations are possible. This motivates the hierarchical implementation of STRAIGHT, as outlined in this section.

Our hierarchical approach progressively zooms in on the updatable regions, thus increasing its discretization density. The process starts with a length map that spans the initial wide location and angle ranges, as described in the previous section. Every time a set of equidistant curves are processed, the rectangular bounding box containing the updatable region (illustrated in the left image of Fig. 8) is upscaled, so that it takes up the complete length map (right image of Fig. 8). This way, although the length map has a constant size, it progressively addresses narrower location and angle ranges around $(\delta_p, \delta_\theta)$, effectively increasing the resolution of the estimates. Since the resolution can increase indefinitely, a coarse discretization of $L(\cdot, \cdot)$ (we use an array of size 21×21) becomes sufficient to obtain long line segments and fewer pixels are tested, thus resulting in computationally efficient line segment extractions. Since, as described in the previous section, a length map may contain multiple disconnected updatable regions, corresponding to different line segments passing through p_0 , this process also divides the length map into multiple ones, each focused on a particular updatable region, and each estimation proceeds



Fig. 8. Illustration of the hierarchical implementation of STRAIGHT. Left: length map, with the bounding box of the updatable region. Right: the same region, after upscaling.

To implement the length map upscaling in the hierarchical STRAIGHT, we use Nearest Neighbor interpolation.

V. EXPERIMENTS

In the absence of an established database for benchmarking the performance of methods for line segment extraction, we single out demonstrative results of STRAIGHT, contrasting them with the ones obtained with the HT [11] and the state-of-the-art LSD [16] (the superiority of LSD when compared to several other methods is thoroughly demonstrated in [16]). We first describe experiments with synthetic images to illustrate extreme cases that help to characterize the general behavior of STRAIGHT. Then, we present results obtained with several real world images that demonstrate its performance in practice.

independently.

A. Synthetic images

We start by illustrating that STRAIGHT succeeds in cases tailored to the HT, *i.e.*, when processing images for which the HT exhibits clear superiority with respect to local methods. We use again the synthetic image used in the Section I, reproduced on the left of Fig. 9. This is a binary image used in a review of several HT-based line segment extraction methods [20]. As we have anticipated in the Section I, and in accordance with the conclusions of [20], the HT succeeds in correctly extracting the lines from this image. In fact, although the multiple crossings make this image visually complex, the HT accumulator array exhibits the desired prominent peaks (see Fig. 1), capturing the fact that the lines are long and not in a very large number. In the third image of Fig. 9, we show the results of LSD. That a pair of twin segments is extracted for each one in the original image is due to the fact that LSD treats the binary image as any other, *i.e.*, as a grey-level one, and both light-to-dark and dark-to-light transitions are detected. However, what is more important is that the local nature of the LSD limits its performance, particularly in resolving the line intersections, making it fail the extraction of several complete segments that cross each other. The rightmost image displays the result of STRAIGHT, showing that it successfully extracts the majority of the line segments, regardless of the intersections (to make the comparison fair, we also processed the image as a grey-level one, originating the double-detection effect).



Fig. 9. Clutterless image with prominent lines. From left to right: original binary image, result of the HT [11], LSD [16], and the proposed method STRAIGHT.

We now illustrate the behavior of the algorithms when dealing with the other extreme of the spectrum, *i.e.*, with images whose line segments are characterized by being frontiers of differently textured regions, rather than abrupt changes in a very smooth intensity level. We use the synthetic images in the left of Fig. 10, which were generated by adding noise to a piecewise constant map. The top image simulates a scenario where a textureless objects occludes a textured one (*e.g.*, a wall in front of a tree) and

the bottom one simulates two textured objects. In both cases, although the line segments that separate regions are perceptually evident, they are not trivially mapped to the edges of the images. In fact, since the textures produce a large number of spurious edge points and the perceptual segments do not guaranteedly produce the corresponding edges, the HT [11] fails to extract them and originates a huge number of false detections, as shown in Fig. 10. Differently, LSD [16] succeeds in interpreting the textures as not forming line segments but only captures parts of the real segments for the top image and almost none for the bottom one. This is due to the local nature of LSD, which makes it sensitive to the missing edge points in the line segments. The rightmost images of Fig. 10 display the results of STRAIGHT, showing that it succeeds in extracting the perceptually relevant lines as forming, in both cases, four complete line segments (the few short segments correspond to accidental connected alignments in the random texture).



Fig. 10. Textured images. From left to right: original image, result of the HT [11], LSD [16], and STRAIGHT.

B. Real images

We start be showing the results obtained with the image used in Section I to clarify the limitations of the HT (Fig. 1). This image is challenging due to its dense packing of line segments of multiple lengths. In the top right image of Fig. 11, we display the results of LSD [16], showing that a subset of the line segments are in fact detected. However, a closer look reveals that those are only the line segments that

do not cross other structures and also that several longer segments are detected as fragmented ones. The results of STRAIGHT are in the two bottom images of Fig. 11. We see that our method succeeds in extracting the vast majority of the line segments in the image (exceptions are those which exhibit very low contrast). The fact that the extracted line segments are complete is particularly evident in the bottom right image, which displays only the line segments that have length greater than 50 pixels.



Fig. 11. Top left: image. Top right: LSD [16]. Bottom left: STRAIGHT. Bottom right: STRAIGHT (longer line segments).

To illustrate how the noise affects the extraction of line segments in real images, we report the results obtained with noisy versions of the same image. Fig. 12 synthesizes the results for two levels of zero-mean white Gaussian noise. We see that, with the increase of the noise level, LSD [16] originates more segment fragmentations and a progressive failure to detect some line segments. The performance of STRAIGHT declines in a less steep way, as expected from its global nature.

Fig. 12. Left: noisy images ($\sigma = 10$ on the top and $\sigma = 20$ on the bottom). Center: LSD [16]. Right: STRAIGHT.

Fig. 13 presents another illustrative case. It was obtained by processing an image containing a complex scene occluded by a net composed of very long line segments that cross multiple times. The result of LSD [16] shows the net broken into short line segments (several sections of the net are not even extracted). On the other hand, our method was able to obtain almost all the complete line segments of the net, even in locations where the background is complex (exceptions are where the net has a very low contrast with respect to the background). The line segments extracted by our method that have length greater than 50 pixels, displayed in the bottom right image of Fig. 13, make this particularly evident.

Finally, Fig. 14 presents results of using STRAIGHT with real images of various kinds. As desired, the vast majority of long line segments are extracted without artificial fragmentation, despite the multiple segment crossings. Also note that, although some of these images have edges that form curves, STRAIGHT succeeds in approximating these sections in a piecewise linear way, *i.e.*, by a sequence of rectilinear line segments.

VI. CONCLUSION

We have presented a new method for line segment extraction, which we call STRAIGHT (Segment exTRAction by connectivity-enforcing Hough Transform). Our method inherits the global accuracy of



Fig. 13. Top left: image. Top right: LSD [16]. Bottom left: STRAIGHT. Bottom right: STRAIGHT (longer line segments).

the HT and overcomes its limitations, particularly those that arise from not taking into account that line segments are *connected* sets of edge points. Our experiments show that STRAIGHT outperforms current methods for line segment extraction in challenging situations, *e.g.*, when dealing with complex images containing several crossing segments.

We end by pointing out that our approach may pave the way to other improvements in HT-like image edge analysis. In fact, as we saw, the HT leads to erroneous votes, which are eliminated by taking point connectivity into account. Thus, the detection of non-rectilinear shapes, *e.g.*, circles, in challenging scenarios, may also benefit from a similar treatment.

ACKNOWLEDGEMENTS

This work was partially supported by FCT, under ISR/IST plurianual funding, through the PIDDAC Program, and grants MODI-PTDC/EEA-ACR/72201/2006 and SFRH/BD/48602/2008.

REFERENCES

- J. Kosecka and W. Zhang, "Video compass," in *Proc. of European Conference on Computer Vision*. Copenhagen, Denmark: Springer-Verlag, 2002, pp. 657–673.
- [2] C. Schmid and A. Zisserman, "Automatic line matching across views," in Proc. of IEEE International Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, 1997, pp. 666–671.
- [3] B. Micusk, H. Wildenauer, and J. Kosecka, "Detection and matching of rectilinear structures," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, USA, 2008, pp. 1–7.
- [4] D. Slater and G. Healey, "3D shape reconstruction by using vanishing points," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 211–217, 1996.
- [5] W. Krüger, "Robust and efficient map-to-image registration with line segments," *Machine Vision and Applications*, vol. 13, pp. 38–50, 2001.
- [6] J. Liu, Y. Chen, and X. Tang, "Decomposition of complex line drawings with hidden lines for 3d planar-faced manifold object reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 3–15, 2011.
- [7] P. Fränti, E. Ageenko, H. Kälviäinen, and S. Kukkonen, "Compression of line drawing images using hough transform for exploiting global dependencies," in *Proc. of the Fourth Joint Conference on Information Sciences*, vol. 4, Research Triangle Park, North Carolina, USA, 1998, pp. 433–436.
- [8] R. Dahyot, "Statistical Hough Transform," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 8, pp. 1502–1509, August 2009.
- [9] A. Borkar, M. Hayes, and M. Smith, "Polar randomized Hough Transform for lane detection using loose constraints of parallel lines," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Prague, Czech Republic, 2011, pp. 1037–1040.
- [10] P. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3.069.654, December 1962.
- [11] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to detect lines and curves in pictures." *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.
- [12] N. Guil, J. Villalba, and E. Zapata, "A fast Hough Transform for segment detection." *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1541–1548, 1995.
- [13] V. Kamat-Sadekar and S. Ganesan, "Complete description of multiple line segments using the Hough Transform," *Image and Vision Computing*, vol. 16, no. 9-10, pp. 597 613, 1998.
- [14] D. Guru, "A simple and robust line detection algorithm based on small eigenvalue analysis," *Pattern Recognition Letters*, vol. 25, no. 1, pp. 1–13, 2004.
- [15] Y. Lee, H. Koo, and C. Jeong, "A straight line detection using Principal Component Analysis," *Pattern Recognition Letters*, vol. 27, pp. 1744–1754, 2006.
- [16] R. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.
- [17] J. Illingworth and J. Kittler, "A survey of efficient Hough Transform methods," in *Proceedings of Alvery Vision Club Conference*, Cambridge, 1987, pp. 319–326.
- [18] H. Li, M. Lavin, and R. Le Master, "Fast Hough Transform: A hierarchical approach," Computer Vision, Graphics, and Image Processing, vol. 36, pp. 139–161, 1986.
- [19] J. Illingworth and J. Kittler, "The adaptive Hough Transform," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 690–698, 1987.

- [20] H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja, "Probabilistic and non-probabilistic Hough Transforms: overview and comparisons," *Image and Vision Computing*, vol. 13, pp. 239–252, 1995.
- [21] A. Desolneux, L. Moisan, and J. Morel, Gestalt theory and image analysis, a probabilistic approach, February 2006.
- [22] M. Fischler and R. Bolles, "RANdom SAmple Consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.
- [23] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, 2004.
- [24] R. Nevatia and K. Babu, "Linear feature extraction and description," *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 257 269, 1980.
- [25] J. Burns, A. Hanson, and E. Riseman, "Extracting straight lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 425–455, 1986.
- [26] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, pp. 97–111, 2007.
- [27] A. Etemadi, "Robust segmentation of edge data," in Proc. of IEEE International Conference on Image Processing and its Applications, Maastricht, The Netherlands, 1992, pp. 311–314.
- [28] O. Faugeras, R. Deriche, H. Mathieu, N. Ayache, and G. Randall, "Parallel image processing," A. Saoudi, M. Nivat, P. Wang, and H. Bunke, Eds. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1992, ch. The depth and motion analysis machine, pp. 143–175.
- [29] L. Wilson, N. Chen, R. Kelley, and J. Birk, "Image feature extraction using diameter limited gradient direction histograms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 228–235, 1979.
- [30] J. Xiao and M. Shah, "Two-frame wide baseline matching," in *Proc. of IEEE International Conference on Computer Vision*, Nice, France, 2003, pp. 603–609.
- [31] K. Arras and R. Siegwart, "Feature extraction and scene interpretation for map-based navigation and map building," in *Proc. of SPIE, Mobile Robotics XII*, vol. 3210, 1997, pp. 42–53.
- [32] R. Guerreiro and P. Aguiar, "Incremental local Hough Transform for line segment extraction," in *Proc. of IEEE International Conference on Image Processing*, Brussels, Belgium, 2011.



Fig. 14. Results of STRAIGHT for several kinds of real images.

DRAFT