FSG: A statistical approach to line detection via fast segments grouping

Iago Suárez[1,3], Enrique Muñoz[1], José M. Buenaposada[2], Luis Baumela[3]

***Abstract*— Line extraction is a preliminary step in various visual robotic tasks performed in low textured scenes such as city and indoor settings. Several efficient line segment detection algorithms such as LSD and EDLines have recently emerged. However, the state of the art segment grouping methods are not robust enough or not amenable for detecting lines in real-time. In this paper we present FSG, a fast and robust line detection algorithm. It is based on two independent components. A proposer that greedily cluster segments suggesting plausible line candidates and a probabilistic model that decides if a group of segments is an actual line. In the experiments we show that our procedure is more robust and faster than the best methods in the literature and achieves state-of-the art performance in a high level robot localization task such as vanishing points detection.**

I. INTRODUCTION

It is well known that state of the art geometric computer vision algorithms fail dramatically in low textured scenes. However, in some settings, like for example in man-made environments, aligned structures such as windows, balconies and doors abound. These structures produce short and disconnected line segment that computer vision algorithms may leverage on to obtain geometric information used for solving fundamental robotic problems such as Simultaneous Localization and Mapping (SLAM) and Vanishing Points (VPs) Detection.

There is a growing interest in using points and lines in Visual Odometry (VO) [5], [9], [13], [16], [26], [27] and SLAM [20], [30]. Some of these algorithms [5], [10], [16], [20] work directly with segments detected with LSD [23] or EDLines [1] and match them using the Line Band Descriptors (LBD) [29] or the Mean–Standard deviation Line Descriptor (MSLD) [24]. Other algorithms try to match full lines by performing segment grouping as a preliminary step [26], [30] with heuristics mainly aimed at achieving a very fast execution.

The input to many VPs estimation algorithms are line segments [12], [14], [25], [28]. Although some of them directly use these segments [18], most of them have a preliminary segment grouping step [12], [14], [25], [28]. State-of-the-art VP estimation algorithms are very slow. Kulger *et al.* [12] takes 45 seconds to process a single image and Lezama *et al.* [14] takes around 30 seconds. Another state-of-the-art method, Zhai *et al.* [28], takes 1 second but requires a GPU. Computing with a GPU could be a problem in battery limited devices such as smartphones or drones.

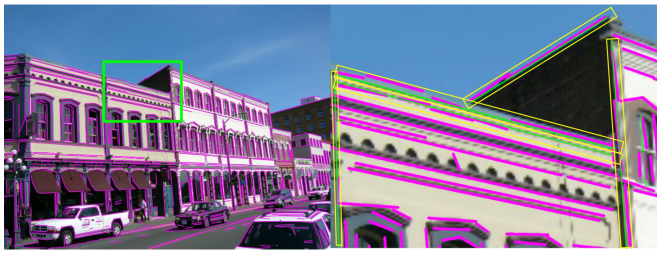
Fig. 1: **Problem statement:** Detected line segments are not usually collinear. **Left:** Line segments detected in an image using LSD (*magenta lines*). **Right:** Detected segments are not strictly aligned with straight scene lines (*green lines*). The smallest bounding box containing the segments of a scene line is plotted in yellow.

Recently some developments in SLAM aim to use simultaneously heterogeneous features with different level of complexity: points, lines, planes and VPs. Lu *et al.* [17] use LSD segments to detect VPs and also to detect full lines with RANSAC. Camposeco *et al.* [6] use also VPs in VO being the inputs the LSD segments and the measurements from an IMU. They remove some steps in LSD to achieve faster execution.

Large and well connected groups of segments produce accurate line estimations that computer vision may leverage on to solve geometric problems in low textured contexts. As far as we know there is no line estimation algorithm that is both fast and robust. In this paper we propose the Fast Segment Grouping (FSG) algorithm that satisfies both requirements. The contributions of our paper are:

1) A very efficient greedy procedure for generating candidate lines from groups of image segments.
2) A probabilistic test to check if a group of segments is an actual line.
3) A segment grouping benchmark built on the York Urban Data-set. We use LSD [23] for segment detection and manually mark segments that belong to full lines. This database is used in our experiments and it is publicly available at `https://github.com/graffter/fsg-benchmark`.

II. PREVIOUS WORK

Classic methods for line segment detection first apply a Canny edge detector [7] followed by a Hough transform [3] or its probabilistic and efficient variant [19]. Recently

[1]The Graffter. Campus Montegancedo s/n. Centro de Empresas. 28223 Pozuelo de Alarcón, Spain. Emails: {iago.suarez, enrique.munoz}@thegraffter.com

[2]ETSII. Universidad Rey Juan Carlos. C/ Tulipán, s/n. 28933 Móstoles, Spain. Email: josemiguel.buenaposada@urjc.es

[3] Departamento de Inteligencia Artificial. Universidad Politécnica de Madrid. Campus Montegancedo s/n. 28660 Boadilla del Monte, Spain. Email: lbaumela@fi.upm.es

LSD [23] and EDLines [1] use local approaches for linking the edge pixels generating isolated segments in a fast and accurate way. However, LSD and EDLines, being local aproaches, extract short segments. A recent approach, MCMLSD [2], finds longer segments by combining the advantages of global probabilistic Hough methods for line detection with spatial analysis in the image domain.

On the other hand, tasks as VP estimation and VO will benefit from the detection of full lines. So, we address the problem by extracting short segments with LSD or EDLines, that are further grouped into full lines. In this section we review the most prominent algorithms for line segment grouping. We organize them into heuristic, clustering, probabilistic, and geometric-based methods.

One of the first fast heuristic methods, Jang *et al.* [11], uses a line segment voting scheme. Segments are assigned to different candidate lines and, using some heuristics, the most likely lines are returned with their associated segments. Zuo *et al.* [30] group two segments if some distances from their middle and end points are small. Yang *et al.* [26] organize the candidates into buckets with similar middle point locations and orientations. They merge segments whose angles and distance are below a threshold.

Clustering-based methods for segment grouping usually work in the line parameter space such as the Hough transform. The work of Bandera *et al.* [4] starts with a Canny edge detection followed by a Randomized Hough Transform for detecting segments. The segments are clustered using the Variable Bandwidth Mean Shift algorithm in the space of line parameters.

The most prominent approaches based on probabilistic models use the *a contrario* methodology [15] to validate a hypothesis based on the expected number of false detections or false alarms. Rajaei *et al.* [21], [22] propose an approach to detect "non-local alignments." Lezama *et al.* [14] use segment end points within an *a contrario* point alignment detector. While this criteria can be valid, they propose candidates by brute force, checking all the possible point pairs in [14] or with an adjacency matrix in [22], both approaches are computationally demanding and, hence, not adequate for real-time settings.

III. FAST SEGMENTS GROUPING (FSG)

Our line detection algorithm takes as its starting point the segments detected in an image. It is based on (1) a probabilistic criteria to accept a group of segments as a line, and (2) a greedy algorithm that proposes clusters of segments as line hypotheses. In this section we present both elements.

A. Probabilistic Segments Group Validation

Let $\mathcal{S}$ be the set of segments detected in an image of n×m pixels and $\mathcal{H}$ a set of s segments randomly distributed in the same image. Let $\mathcal{C}_s$ be a group of c segments from $\mathcal{S}$ and B the smallest bounding box that encloses all segments in $\mathcal{C}_s$.

Let $\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s)$ be the expected number of boxes of size equal or smaller than B that enclose c segments of the same length as those in $\mathcal{C}_s$ in $\mathcal{H}$. We accept the group of segments

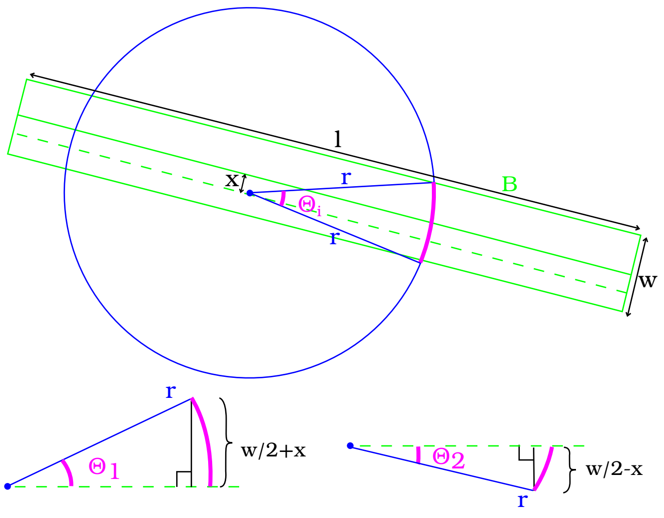

Fig. 2: **Probability that a random segment falls in the Box:** Statistical method to calculate the probability that the segments in $\mathcal{C}_h$ are bounded by B ($P[\mathcal{C}_h \in B]$).

$\mathcal{C}_s$ as a line if $\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s)$ is below a certain threshold, ϵ. The intuition behind this criterion is simple. If the segments in $\mathcal{C}_s$ are well aligned then B will be very thin and the chances that a set of c segments from $\mathcal{H}$ fall into B will be very small.

We assume that the segment's endpoints position in $\mathcal{H}$ follow a uniform distribution, $\mathcal{U}(n, m)$. Under this assumption,

$$\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s) = \binom{s}{c} P[\mathcal{C}_h \in B], \tag{1}$$

where $\mathcal{C}_h$ is a set of segments in $\mathcal{H}$ with the same cardinality and lengths as those in $\mathcal{C}_s$ and $P[\mathcal{C}_h \in B]$ is the probability that the segments in $\mathcal{C}_h$ are bounded by B. This probability is given by:

$$P[\mathcal{C}_h \in B] = \prod_{i=0}^{s} P[b(s_i)] \cdot P[e(s_i)|b(s_i), l(s_i)], \tag{2}$$

where s_i is the i-th segment in $\mathcal{C}_h$, $b(s_i)$ and $e(s_i)$ are, respectively, the beginning and ending points of segment s_i, $P[b(s_i)] = \frac{w \cdot l}{n \cdot m}$ is the probability of the first point of segment s_i falling into a box B of size $w \times l$ and $P[e(s_i)|b(s_i), l(s_i)]$ is the probability of $e(s_i)$ falling into B given that $b(s_i)$ is already in B and segment s_i has length $l(s_i)$. For simplicity, if we assume that B has infinite length, then

$$P[e(s_i)|b(s_i), l(s_i)] = \frac{\bar{\theta}_i}{\pi},$$

where $\bar{\theta}_i$ is the average angle of a sector within B with radius $l(s_i)$ and subtended by B (see Fig. 2).

Finally, $\bar{\theta}_i$ is given by

$$\bar{\theta}_i = \frac{1}{w/2} \int_0^{w/2} (g_1(x) + g_2(x)) dx, \tag{3}$$

where (see Fig. 2)

$$g_1(x) = \arcsin\left(\min\left(1, \frac{w/2 + x}{l(s_i)}\right)\right)$$

and

$$g_2(x) = \arcsin\left(\min\left(1, \frac{w/2 - x}{l(s_i)}\right)\right).$$

Hence, we accept as a good line the set of segments $\mathcal{C}_s$ such that $\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s) < \epsilon$. The acceptance threshold ϵ may be established by cross-validation.

Once we have this criterion, we could find the lines in an image by just enumerating all possible groups of segments $\mathcal{C}_s$ and accept as lines the non-overlapping set of groups with smallest $\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s)$. However, this would be computationally very demanding, preventing our approach from being used in real robotics applications. To speed this process, in the following section we introduce an efficient group proposer.

B. Segments Groups Proposer

The proposer is based on the fact that line segment detectors [1], [23] produce segments with some error in the location of endpoints caused by glitters, shadows, occlusions, etc. Our algorithm starts with the endpoints of the longest segments (the most stable ones) and greedily tries to enlarge the group by including new segments that fall inside a cone defined by the uncertainty in the location of the group end points (See Fig. 3). For improving the efficiency candidate segments are organized by length and orientation (see algorithm 1).

The input to the algorithm is a set of segments, $\mathcal{S}$, with $|\mathcal{S}| = n$. The first step is to build a partially ordered list, L, of the segments in $\mathcal{S}$ by length. The second step is to build a histogram of segment orientations P_θ. Both of these steps have complexity $O(n)$. Next, the greedy procedure begins traversing L from longest to shortest segment:

1) For each $s_i \in L$, if it is not already in a group, using P_θ find the set of similar orientation segments, N.
2) Initialize a new candidate group of segments with only one segment, $\mathcal{C}_s = \{s_i\}$.
3) We define a circle of uncertainty on each of the endpoints of group $\mathcal{C}_s$ that delineate a cone where candidates must lie (see purple lines in Fig. 3). This cone is defined by two lines l_1 and l_2 calculated with the function computeTangentLines($\mathcal{C}_s$) (see line 11 in algorithm 1).
4) Each segment, s_k, in N is checked in turn as a candidate to be added to $\mathcal{C}_s$. First of all a fast test is performed over the middle point of s_k, $m(s_k)$ to check if the segment is in the cone ($D_1[j]$ and $D_2[j]$ have the same sign in line 17 of algorithm 1). Then we use the probabilistic test $\mathbb{E}_H(\mathcal{S}, (\mathcal{C}_s + \{s_k\})$ to include it in $\mathcal{C}_s$.
5) Go to step 1) to process the next segment in L.

The partial ordering of segments, the histogram of orientations and the fast cone-shaped region check are the cornerstones of the greedy algorithm. In the experiments we show that it is both fast and accurate.

IV. EXPERIMENTS

We evaluate the proposed FSG approach by using quantitative comparisons to the state of the art. We accordingly introduce a ground truth data-set to specifically evaluate line segment grouping methods. Also, we provide statistical validation, and a comparison of FSG with the state of art for estimating the vanishing points of an image.

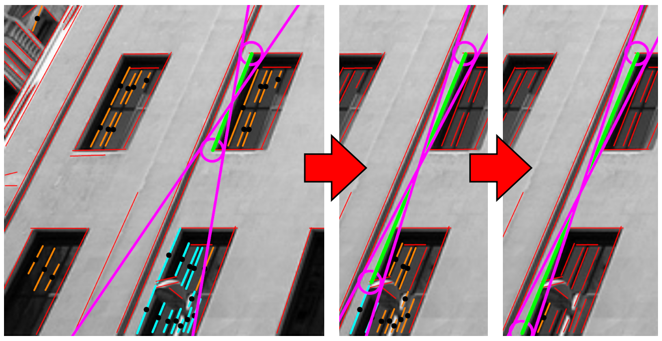
Fig. 3: **Segments groups proposer:** Greedy process of segments proposing based on the endpoint error circles. Purple lines l_1 and l_2 define the cone-shaped search region, red segments are discarded by orientation, orange are discarded because being out of the search region, blues are the candidates and green are the already selected as part of the base segment.

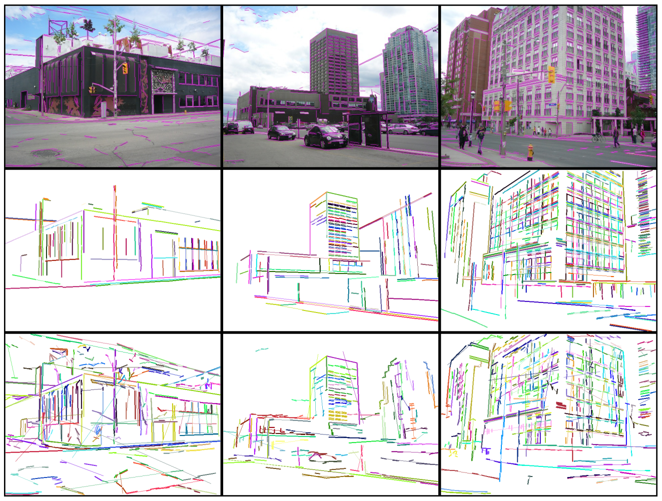
Fig. 4: **Ground truth data-set for segment grouping:** We annotate the YUD with positive and negative examples of image segments clusters. **Top:** Some images from the YUD with line segments detected using LSD (*magenta lines*). **Middle:** Positive clusters, i.e. image segments that belong to straight lines in the scene. **Bottom:** Negative clusters, i.e. segment groups that are not collinear in the scene. We plot segments in the same cluster with the same color. Also, we plot a line using this color between the cluster extrema.

A. Ground truth database generation

To the best of our knowledge, there are no specific datasets for evaluating line segments clustering methods. To that purpose, we augment a well-established data-set, the York Urban Database (YUD) [8]. It comprises a set of images of buildings, each with its camera orientation and calibration.

Algorithm 1 Greedy Groups Proposer Pseudo-code

Input: $\mathcal{S}$ (LSD Segments), ϵ

Output: S^* (the set of segments groups)

```
1:  $S^* = \emptyset$
2:  $L \leftarrow$ semiSortByLength($\mathcal{S}$)
3:  $P_\theta \leftarrow$ createOrientationHistogram($\mathcal{S}$)
4:  for all $s_i \in L$ do
5:     if $s_i \in S^*$ then Continue end if
6:     $\mathcal{C}_s = \{s_i\}$
7:     $N \leftarrow$ getSegmentsInNearestBins($P_\theta$, $s_i$)
8:     $M \leftarrow$ getMiddlePoints($N$)
9:     searching = true
10:    while searching do
11:       $l_1$, $l_2 \leftarrow$ computeTangentLines($\mathcal{C}_s$)
12:       if $l_1^T \cdot b(\mathcal{C}_s) < 0 \neq l_2^T \cdot b(\mathcal{C}_s) < 0$ then $l_1 = -l_1$
          end if
13:       $D_1 = l_1^T \cdot M$ ; $D_2 = l_2^T \cdot M$
14:       BestSeg = $\emptyset$ ; BestSegA = MaxVal
15:       $M_{new} = \emptyset$ ; $N_{new} = \emptyset$
16:       for j = 0 to $|N|$ do
17:          if sameSign($D_1[j]$, $D_2[j]$) then
18:             $M_{new} = M_{new} + M[j]$
19:             $N_{new} = N_{new} + N[j]$
20:             $A = \mathbb{E}_H(\mathcal{S}, \mathcal{C}_s + \{N[j]\})$
21:             if ($A < \epsilon$) and ($A <$ BestSegA) then
22:                BestSegA = $A$
23:                BestSeg = $N[j]$
24:             end if
25:          end if
26:       end for
27:       if BestSeg = $\emptyset$ then
28:          searching = false
29:       else
30:          $M = M_{new}$ ; $N = N_{new}$
31:          $\mathcal{C}_s = \mathcal{C}_s + \{BestSeg\}$
32:          Delete BestSeg from $M$, $N$ and $P_\theta$
33:       end if
34:    end while
35:    $S^* = S^* + \{\mathcal{C}_s\}$
36: end for
```

It also provides a few lines in each image, to estimate the Manhattan frame relative to the camera (see [8] for further details).

In our annotation we cluster segments that belong to straight structures in the scene into longer, meaningful, lines. Thus, for each image we detect segments using LSD (actually, its OpenCV[1] implementation, with `LSD_REFINE_ADV` enabled). Fig. 4 shows the detected line segments in some images.

We have developed a C++ application that allows us to cluster the detected segments into straight lines. To this end the user picks each segment in an interactive manner, so that segments belonging to distinctive parts of the scene can be easily grouped. Fig. 4 shows some sample segment clusters. We have mainly grouped segments corresponding to long lines of the scene (e.g., buildings) leaving out non-collinear detections (e.g. cars, people, or similar items in the scene).

Furthermore, in order to provide a statistical validation of our method, we also annotate a set of negative examples. Thus, instead of grouping segments belonging to straight lines in the scene, we intentionally cluster segments using the opposite criterion, i.e., manually select segments that are not strictly collinear. Fig. 4 shows some segments that have been clustered as negative examples. Some of them do not even belong to the same structure in the scene.

B. Validation of segment clusters

We approach the validation of a segment cluster as a classification binary problem where the clustered segments can be aligned (POSITIVE Label) or not (NEGATIVE Label). In this experiment our results are compared with those in [15]. Note that since this method is based on a point alignment detector, we define that a segment cluster is labeled POSITIVE if the method returns an alignment and at least half of the points contained in that alignment come from some ground truth labeled segment.

We have generated the ROC curve (see Fig. 6) by changing the rarity threshold of both algorithms. We generate Lezama's *et al.* [15] curve varying ϵ in the interval $[0, 10^7]$. We were not able to evaluate larger values because of the computational cost. In our method we fix the $\epsilon = 1$ and vary the number of segments s in $\mathcal{H}$ as rarity threshold.

Fig. 6 shows that our statistical criteria significantly outperforms Lezama's *et al.* [15] *a contrario* model. The main reason is that our method considers entire segments instead of alignments of the segments endpoints. As the two endpoints of every segment are used independently, accidental alignments of points can be found in Lezama's method (see bottom row in Fig. 5).

In Fig. 5 we show our results compared with Lezama *et al.* [15] for segment grouping. Our method works fine in indoor and outdoor environments robustly detecting all the meaningful lines in the image. Nevertheless, Lezama *et al.* detects some non existent lines and miss other important ones.

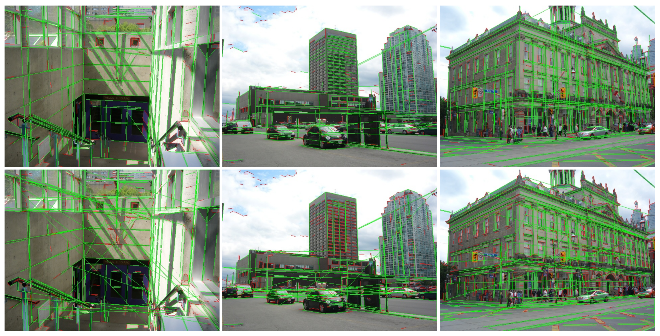

Fig. 5: **Top**: FSG versus **Bottom**: Lezama *et al.* [15]. Comparison of the lines generated from the clustered segments.

[1] https://docs.opencv.org/3.4.1/db/d73classcv_1_1LineSegmentDetector.html

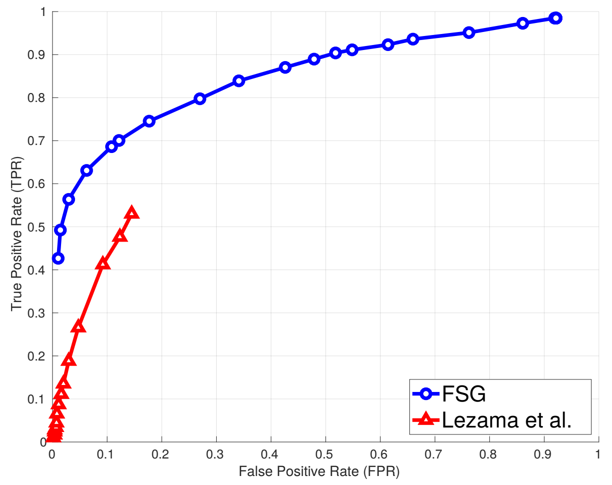

Fig. 6: **Line segment cluster classification**: ROC curve where we compare how well our statistical criteria performs compared to Lezama *et al.* [15] for the first 10 images labeled in the York Urban Database.

C. Vanishing Point estimation

We also compare FSG with other line detectors applying them to a high-level task, such as estimating the VPs of a scene. To this end we compare our approach using Lezama *et al.*'s framework [14] that, to the best of our knowledge, is the state of the art VPs detector. Lezama's algorithm consists of three main steps: 1) detect line segments with LSD; 2) after some heuristic filtering, feed the *a contrario* model with these segments to estimate full lines (i.e. segments groups); and 3) compute the VPs. In Fig. 7 we show the VPs estimation results obtained when feeding step 3) in Lezama's algorithm with image lines computed with five approaches:

- **ground-truth**: using annotated ground truth lines;
- **Lezama *et al.*** [14]: original approach in [14], including the heuristic segment filter (LSD + heuristic segment filter + *a contrario* decision);
- **Lezama *et al.* (basic)**: approach in [14], with no heuristic segment filter (LSD + *a contrario* decision);
- **MCMLSD**: segments from the method in [2];
- **FSG**: segments resulting from our method, (LSD + FSG).

The **ground-truth** approach provides our experimentation with an upper bound on VP estimation. We also include **Lezama *et al.* (basic)** to evaluate the impact of the filtering heuristics in the results of this approach.

Fig. 7 shows the cumulative histograms of the horizon line detection error for each approach. The horizon line error is computed as the Euclidean distance between the extremes of the estimated and ground-truth horizon line segments, weighted by the image height. We provide an overall score, that can be quantitatively compared, by measuring the area under the curve (AUC) of the cumulative histogram curves.

In terms of the AUC score, all three approaches provide similar results, very close to the upper bound provided by the VPs estimated with the ground truth lines. FSG performs marginally better than [14]. Meanwhile, MCMLSD also performs marginally better than FSG, because of its global approach.

Local Segment Detectors		Segments Grouping Algorithms		Global Segment Detectors	
LSD	**EDLines**	**FSG**	**Lezama et al.**	**PPHT**	**MCMLSD**
32	6	6	14961	22	4686

TABLE I: **Average execution times for line segment detection (ms) in the York Urban Dataset.**

Table I shows the average execution times for various algorithms measured on an Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz with 16 GB of RAM. The fastest algorithm in the global segment detector group is PPHT, the OpenCV Probabilistic Hough Line Transform [19] with a Canny edge detector [7]. However it has some important drawbacks (see [23]). The alternative, MCMLSD [2] is very accurate, however, its computational cost prevents it from being used in a real-time robotic system.

Similarly, if we analyze the segment grouping algorithms, the computational cost of Lezama *et al.* [14] is so high, that it cannot be used in real-time, no matter what local segment detector is used. The computational cost of FSG is three orders of magnitude smaller than its competitor. The fastest configuration, EDLines [1] + FSG, could be used to extract image lines at 83 frames per second, in the computer that we used for our experiments.

Finally, from the results in Fig. 7 and Table I, we can conclude that LSD + FSG has an accuracy comparable to the state of the art, but with an execution speed three orders of magnitude faster.

V. CONCLUSIONS

In this work we present FSG, an accurate real-time line detection algorithm based on grouping line segments. FSG offers: 1) a very efficient greedy segment group candidate proposer, and 2) a statistical validation criteria to accept a group of segments as a line.

To experimentally evaluate our statistical validation criteria we propose a new data set generated by augmenting the York Urban Database (YUD) [8] with new line labels. With this data set we have experimentally proved that our validation criteria is better than the state-of-the-art 2D *a contrario* point alignment algorithm [15].

We have compared the accuracy of VPs estimated with an LSD segment detector and FSG line extractor with the best in the literature. The LSD + FSG approach achieved results comparable to the state-of-the-art techniques in terms of accuracy in the estimation of the horizon line, but with an execution speed three orders of magnitude faster.

So, the proposed segment grouping approach, a seemingly small element in a robotic system, may have a big impact in the overall performance of higher level robotics task, such as VO through VPs estimation.

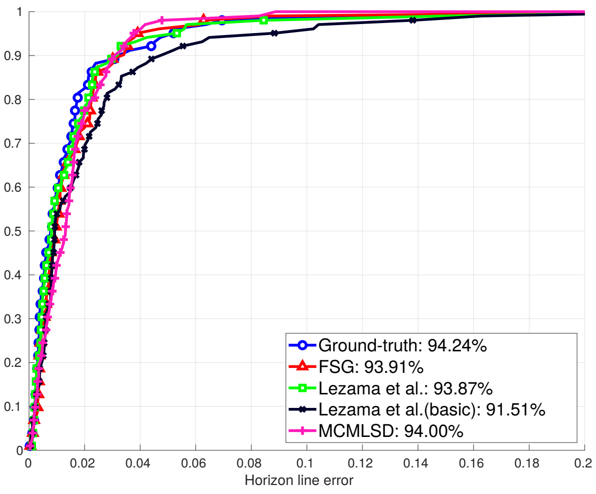

Fig. 7: **Horizon line error:** Cumulative histogram plots between the proposed approach (*red*) and [14](*green*). Upper and lower bounds are provided by using the manually annotated data from YUD (*blue*) a basic, heuristics-free, version of [14](*black*). Each method has been annotated with its AUC score (see legend).

ACKNOWLEDGMENTS

The following funding is gratefully acknowledged. I. Suárez Doctorado Industrial DI-16-08966, E. Muñoz Torres Quevedo PTQ-15-07946, J.M. Buenaposada and L. Baumela MINECO project TIN2016-75982-C2-2-R, The Graffter NEOTEC SNEO-20161065.

REFERENCES

[1] Cuneyt Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.

[2] Emilio J Almazan, Ron Tal, Yiming Qian, and James H Elder. Mcmlsd: A dynamic programming approach to line segment detection. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5854–5862. IEEE, 2017.

[3] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. In *Readings in computer vision*, pages 714–725. Elsevier, 1987.

[4] Antonio Bandera, José Manuel Pérez-Lorenzo, Juan Pedro Bandera, and F Sandoval. Mean shift based clustering of hough domain for fast line segment detection. *Pattern Recognition Letters*, 27(6):578–586, 2006.

[5] Suman Raj Bista, Paolo Robuffo Giordano, and François Chaumette. Combining line segments and points for appearance-based indoor navigation by image based visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'17*, 2017.

[6] Federico Camposeco and Marc Pollefeys. Using vanishing points to improve visual-inertial odometry. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5219–5225. IEEE, 2015.

[7] John Canny. A computational approach to edge detection. In *Readings in Computer Vision*, pages 184–203. Elsevier, 1987.

[8] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European conference on computer vision*, pages 197–210. Springer, 2008.

[9] Ruben Gomez-Ojeda, Jesus Briales, and Javier Gonzalez-Jimenez. Pl-svo: Semi-direct monocular visual odometry by combining points and line segments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4211–4216. IEEE, 2016.

[10] Ruben Gomez-Ojeda and Javier Gonzalez-Jimenez. Robust stereo visual odometry through a probabilistic combination of points and line segments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2521–2526. IEEE, 2016.

[11] Jeong-Hun Jang and Ki-Sang Hong. Fast line segment grouping method for finding globally more favorable line segments. *Pattern Recognition*, 35(10):2235–2247, 2002.

[12] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. In *German Conference on Pattern Recognition, GCPR*, pages 17–28. Springer, 2017.

[13] Thomas Koletschka, Luis Puig, and Kostas Daniilidis. Mevo: Multi-environment stereo visual odometry. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4981–4988. IEEE, 2014.

[14] José Lezama, Rafael Grompone von Gioi, Gregory Randall, and Jean-Michel Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 509–515, 2014.

[15] José Lezama, Jean-Michel Morel, Gregory Randall, and Rafael Grompone Von Gioi. A contrario 2d point alignment detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):499–512, 2015.

[16] Yan Lu and Dezhen Song. Robust rgb-d odometry using point and line features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3934–3942, 2015.

[17] Yan Lu and Dezhen Song. Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. *IEEE Transactions on Robotics*, 31(3):736–749, 2015.

[18] Adam Herout (Brno University of Technology) Markéta Dubská (Brno University of Technology). Real projective plane mapping for detection of orthogonal vanishing points. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[19] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[20] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: real-time monocular visual slam with points and lines. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4503–4508. IEEE, 2017.

[21] B. Rajaei, R. G. von Gioi, and J. M. Morel. From line segments to more organized gestalts. In *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 137–140, March 2016.

[22] Boshra Rajaei and Rafael Grompone von Gioi. Gestaltic grouping of line segments. *Image Processing On Line, IPOL (preprint)*, 2017.

[23] 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*, 32(4):722–732, April 2010.

[24] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009.

[25] Yiliang Xu, Sangmin Oh, and Anthony Hoogs. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1376–1383. IEEE, 2013.

[26] Shichao Yang and Sebastian Scherer. Direct monocular odometry using points and lines. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3871–3877. IEEE, 2017.

[27] Hongsheng Yu and Anastasios I Mourikis. Vision-aided inertial navigation with line features and a rolling-shutter camera. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 892–899. IEEE, 2015.

[28] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattanworld. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 5657–5665. IEEE, 2016.

[29] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013.

[30] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual slam with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1775–1782, 2017.