The multiscale line segment detector

Yohann Salaün[1,2], Renaud Marlet[1] and Pascal Monasse[1]

[1] LIGM, UMR 8049, École des Ponts, UPE, Champs-sur-Marne, France
[2] CentraleSupélec, Châtenay-Malabry, France
{yohann.salaun, renaud.marlet, pascal.monasse}@enpc.fr

Abstract. We propose a multiscale extension of a well-known line segment detector, LSD. We show that its multiscale nature makes it much less susceptible to over-segmentation and more robust to low contrast and less sensitive to noise, while keeping the parameter-less advantage of LSD and still being fast. We also present here a dense gradient filter that disregards regions in which lines are likely to be irrelevant. As it reduces line mismatches, this filter improves the robustness of the application to structure-from-motion. It also yields a faster detection.

1 Introduction

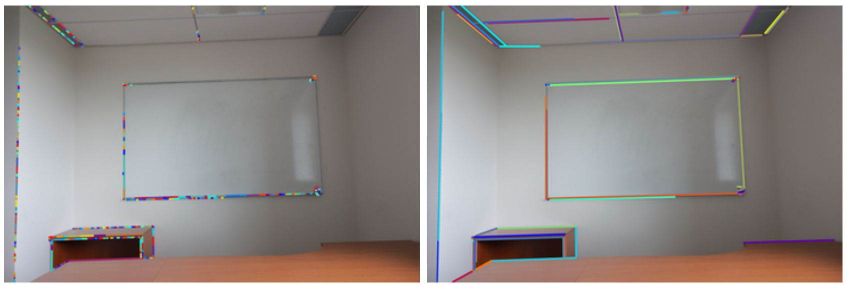


Fig. 1: Lines detected with LSD [2] (left) or with MLSD [3] (right). The picture has a resolution of 15 Mpixels.

Among proposed line detectors, LSD [2] is one of the best and most popular methods. It accurately detects segments and does not use any threshold tuning, relying instead on the *a contrario* methodology. Though results are very good for small images (up to 1 Mpixel), it tends to give poorer results with high resolution images (5 Mpixels and more). As explained by Grompone von Gioi et al. [2], the detection is different after scaling or croping the picture. For high resolution images, detections are often over-segmented into small bits of segments and some lines are not even detected (see Fig. 1).

These poor results can be traced back to the greedy nature of LSD. Detected segments are in fact rectangular areas that contain a connected cluster of pixels with gradients that are similarly oriented. After they are identified, a score representing a number of false alarms (NFA) validates the detection as an actual segment or not. However, in high resolution cases, edges tend to be less strong which breaks the connectivity between pixel clusters and yields over-segmentation or lack of detection in low-contrast areas.

We propose a method that generalizes LSD to any kind of images, without being affected by their resolution. For this, we use a multiscale framework and information from coarser scales to better detect segments at finer scales. In our companion paper [3], we compare it to other state-of-the-art line detectors, namely LSD [2] and EDLines [1], as a building block of a structure from motion (SfM) pipeline [4] to obtain quantitative, objective results.

2 Notation

We use the same notations as the companion paper [3], recalled here. In the following, the k index will denote the scale associated with the feature.

2.1 Upscaled segment

Given a coarse segment s_i^{k-1} of direction $\theta(s_i^{k-1})$ detected with some angular tolerance $\pi\, p_i^{k-1}$ ($0 \leq p_i^{k-1} \leq 1$ represents a probability), we define $\mathcal{A}_i^k$ as the rectangular area of s_i^{k-1} upscaled in I^k, and $\mathcal{P}_i^k$ as the subset of pixels in $\mathcal{A}_i^k$ that have the same direction as s_i^{k-1} up to $\pi\, p_i^{k-1}$:

$$\mathcal{P}_i^k = \left\{ q \in \mathcal{A}_i^k \text{ s.t. } |\theta(q) - \theta(s_i^{k-1})|_{(\text{mod } \pi)} < \pi\, p_i^{k-1} \right\}. \tag{1}$$

where $\theta(q)$ is the direction orthogonal to the gradient at pixel q. Note that we only consider a gradient direction if the gradient magnitude is above a given threshold $\rho = 2/\sin(45°/2)$ as in the original LSD because it is a good trade-off between good and fast detections.

2.2 Fusion score

Given n segments $S = \{s_1, ..., s_n\}$, let $Seg(\cup_{i=1}^n s_i)$ be the best segment computed from the union of the clusters s_i, defined as the smallest rectangle that contains the rectangles associated to all segments s_i. The corresponding fusion score of the set of segments is defined as:

$$\mathcal{F}(s_1, ..., s_n) = \log\left(\frac{\text{NFA}_{\mathcal{M}}(s_1, ..., s_n, p)}{\text{NFA}_{\mathcal{M}}(Seg(\cup_{i=1}^n s_i), p)}\right). \tag{2}$$

The NFA is computed with equation (3) of the companion paper [3]:

$$\text{NFA}_{\mathcal{M}}(S, p) = \gamma N_L \binom{(NM)^{\frac{5}{2}}}{n} \prod_{i=1}^{n} (|s_i| + 1)\mathcal{B}(|s_i|, k_{s_i}, p) \tag{3}$$

in an $N \times M$ image, where γ is the number of tested values for the probability p, N_L the number of possible segments in the image, and k_s the number of pixels in the rectangle aligned with its direction, with tolerance πp. It uses the tail of the binomial law:

$$\mathcal{B}(|s|, k_s, p) = \sum_{j=k_s}^{|s|} \binom{|s|}{j} p^j (1-p)^{|s|-j}. \tag{4}$$

The fusion score defines a criterion for segment merging that does not rely on any parameter. If positive, the segments $s_1, ..., s_n$ should be merged into $\mathcal{S}eg(\cup_{i=1}^{n} s_i)$ otherwise they should be kept separate.

3 Dense-gradient filter

For SfM purpose, a too high density of segment detections in some area, such as a grid pattern, often leads to incorrect results for line matching. The density of similar lines also leads to less accurate calibration because it weighs too much similar information and thus tends to reduce or ignore information from lines located in other parts of the image. To address this issue, we designed a filter that disables detections in regions with too dense gradients. It also allows a faster detection as these regions often generate many tiny aligned segments that would need to be merged during our post-detection merging.

For this, we first detect regions with a local gradient density above a given threshold. The segment detection is then disabled in these areas. The process is fast because we apply it only at the coarsest scale using summed area tables.

With this filter, we may obtain a less exhaustive segment detection in these areas and at their borders. However, it leads to a better matching and a better calibration. It also decreases computation time for images with this type of regions.

4 Implementation

Our implementation is available on GitHub (`https://github.com/ySalaun/MLSD`).

4.1 Main algorithm

Our algorithm consists in an iterative loop of three steps for each considered scale of the picture:

1 **Multiscale transition:** Upscale information from previous, coarser scale and use this information to compute segments at current, finer scale.
2 **Detection:** Detect segments using the standard LSD algorithm [2].
3 **Post-detection merging:** Merge neighboring segments at current scale.

```
Input: Image $I$
Output: Set of segments $\mathcal{S}$

for $k = 0$ to $K$ do
    Compute downscaled image $I^k$ ($I^K$ is the original image)

    // 1. Initialize segment set at this scale from previous scale information, if any

    if $k = 0$ then
        $\mathcal{S}^k \leftarrow \emptyset$
        Optionally, disable detection in some regions with dense-gradient filter
    else
        $\mathcal{S}^k \leftarrow$ UPSCALE($\mathcal{S}^{k-1}$)
    end if

    // 2. Add the segments detected with LSD [2]:
    $\mathcal{S}^k \leftarrow \mathcal{S}^k \cup$ LSD($I^k$)

    // 3. Merge aligned neighbors
    $\mathcal{S}^k \leftarrow$ MERGE($\mathcal{S}^k$)
end for

return $\mathcal{S}^K$
```

Fig. 2: Multiscale Line Segment Detector (MLSD).

Note that step 2 uses the exact same procedure as LSD and thus will not be described in this paper. The dense-gradient filter can optionally be used at the coarsest scale.

The number of considered scales is noted K. Though it can be chosen by the user, we compute it automatically depending on the size of the picture:

$$K = \min\{k \in \mathbb{N} \text{ s.t. } \max(w, h) \leq 2^k s_{max}\},$$

where w (resp. h) is the width (resp. height) of the picture. We use a scale step of 2 and chose $s_{max} = 1000$ as we did not observe over-segmentation for images of size lower than 1000×1000 pixels and reducing too much the picture size can create artifacts.

The overall algorithm is described in Fig. 2. The successive steps are described below with a pseudo-code giving the main steps and additional details in the text.

4.2 Dense-gradient filter

The general idea of the dense-gradient filter is to discard from detection areas in which there is a high density of pixels with strong gradients.

Experimentally, we observed that it is difficult to set a density threshold for a proper filtering. If the density threshold is too low, it tends to discard pixels that may belong to interesting segments. If it is too high, it does not filter out enough pixels. For this reason, we perform the filtering in two steps. First, we identify which pixels are at the center of dense-gradient areas. Second, we disregard a region which is larger than the one that is used to evaluate the density of strong gradients.

```
**Input:** Image at coarsest scale $I^0$
**Output:** Mask $\tilde{I}^0$ of valid pixels
Compute summed area table of pixels with significant gradient magnitude for $I^0$
$\tilde{I}^0 \leftarrow$valid
for all pixel $p \in I^0$ do
  Compute density $\tau$ of pixels with significant gradient near $p$
  if $\tau > \tau_{DENSE}$ then
    Invalidate pixel $p$ in $\tilde{I}^0$
  end if
end for
Expand invalidated pixels in $\tilde{I}^0$ by dilation
return $\tilde{I}^0$
```

Fig. 3: Dense gradient filter.

This procedure is implemented in our code with the function *denseGradient-Filter* and described in Fig. 3. It consists in 3 steps:

1 **Summed area table:** We use a value of 1 for pixels with a gradient above ρ and 0 otherwise.
2 **Filtering:** For each pixel, we estimate the local density τ of pixels within a 5×5 window. The filtered pixels are those with a density $\tau > \tau_{DENSE} = 0.75$.
3 **Expansion:** For each filtered pixels p, we also discard every pixel inside a 21×21 window centered at p.

4.3 Multiscale transition

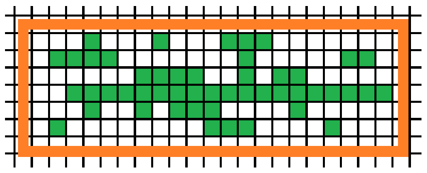
We use a multiscale exploration that propagates detection information at coarse scales to finer scales, which contributes in reducing over-segmentation.

This procedure is implemented in our code with the function *refineRawSegments*, described in Fig. 4 and some parts are illustrated in Fig. 5. It iterates over each segment detected at the previous scale:

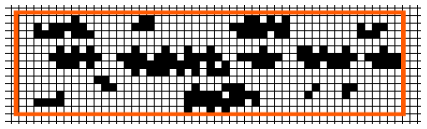
1 **Upscaling:** The segment coordinates are upscaled and we compute the set $\mathcal{P}_i^k$ of pixels aligned with the segment direction (1).
2 **Aggregation:** Pixels inside $\mathcal{P}_i^k$ initialize clusters and are aggregated following an 8-neighborhood greedy method.
3 **Merging:** The clusters found at step 2 are merged according to the fusion score (2). As we cannot practically consider all the possible groups of clusters, we sort them by increasing NFA (i.e., decreasing meaningfulness) and queue

```
**Input:** Set of segments $\mathcal{S}^{k-1}$ detected at former scale
**Output:** Set of upscaled segments $\mathcal{S}^k$

$\mathcal{S}^k \leftarrow \emptyset$
**for all** $s_i \in \mathcal{S}^{k-1}$ **do**
    1. Upscale $s_i$ coordinates and select in this area the pixels whose gradient direction is similar to the orthogonal direction to $s_i$ (i.e., compute $\mathcal{P}_i^k$)
    2. Aggregate pixels inside $\mathcal{P}_i^k$ into 8-connected components
    3. Merge w.r.t. fusion score (2)
    4. Add to $\mathcal{S}^k$ the resulting segments, if meaningful enough
**end for**

**return** $\mathcal{S}^k$
```

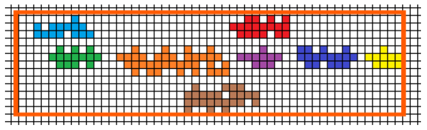
Fig. 4: Multiscale transition.



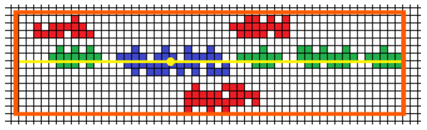
Coarse segment



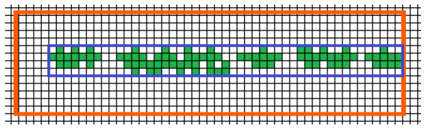
Step 1: Upscaling



Step 2: Aggregation



Step 3: Merging



Step 4: Segment computation

Fig. 5: Illustration of the multiscale processing steps. The large orange rectangular box represent the upscaled region of the coarser segment. The blue rectangular box at step 4 represents the region of the detected segment at the finer scale. $\mathcal{P}_i^k$ is represented at step 1 with black pixels. After aggregation we only keep regions with at least 10 pixels. At step 3, we treat each cluster as an LSD segment with a barycenter, width and direction. We then consider the line that goes through its barycenter and with its direction (represented in yellow) to find merge candidates.

them. Then, for each cluster c, we find the set of clusters that intersects with the line corresponding to c and try to merge the whole set. If merging is validated by the fusion score, we add the new segment inside the queue and dequeue the merged segments.

4 **Segment computation:** For each resulting cluster, if the NFA is low enough, we compute its corresponding segment and add it to the current set $\mathcal{S}^k$.

In the case where no pixel is selected at step 1 ($\mathcal{P}_i^k = \emptyset$) or no segment is added at step 4, we add the original segment into $\mathcal{S}^k$ with a scale information. It is kept as is, unrefined until all scales are explored. This allows detecting segments in low-contrast areas. Following LSD [2], we use a threshold $\epsilon = 1$ for NFA which corresponds to one false detection per image. LSD authors have shown that with values of TODO, the results do not change too much. In the code, this value is represented by the variable *logeps*.

4.4 Post-detection merging

As new segments may be detected by LSD at the current scale, in addition to the segments originating from coarser scales, and as these segments may correspond to some form of over-segmentation, we apply another pass of segment merging, similar to the one used in multiscale transition but simplified to consider a reduced number of possible fusions.

```
Input: Set of segments $\mathcal{S}^k$ detected at current scale
Output: Set of merged segments $\tilde{\mathcal{S}}^k$
for all $s_i \in \mathcal{S}^k$ do
    1. Find aligned neighbors
    2. Possibly merge them into a single new segment w.r.t. fusion score (2)
    3. Add the new segment, if any, in $\mathcal{S}^k$
end for
return $\mathcal{S}^k$
```

Fig. 6: Post-detection merging.

This procedure is implemented in our code with the function *mergeSegments* and described in Fig. 6. It iterates on each segment previously detected. For this, as above, we first sort them by increasing NFA and push them to a queue. We then iterate the following steps until the queue is empty:

1 **Clustering:** For each segment s_i, we consider its central line as in multiscale transition (Sect. 4.3). For each of the two directions of the line, we examine the first cluster intersecting the line and such that its direction is similar to the direction of s_i up to tolerance $\pi\, p_i$.

2 **Merging:** The previously selected segments are merged if needed using the fusion score as a reference (2).

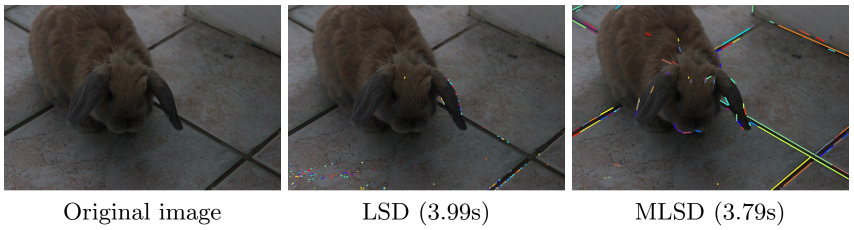


Fig. 7: Lines detected with LSD [2] (middle) and with MLSD [3] (right) in a 15 Mpixels image (left).

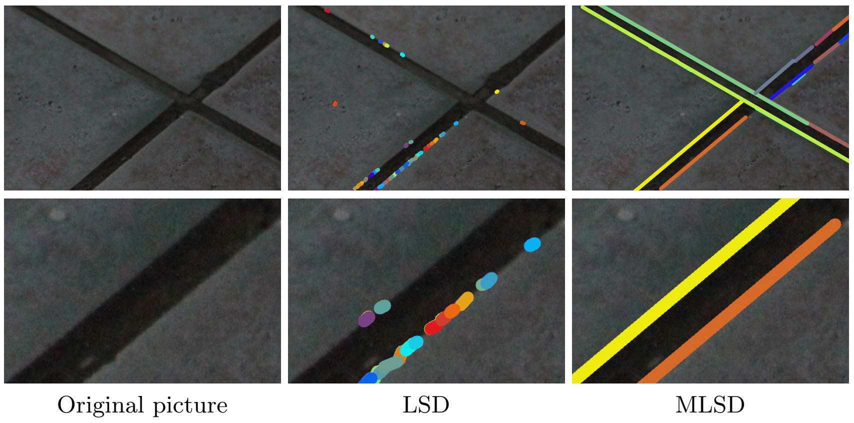


Fig. 8: Zoom of the bottom-right corner of pictures in Fig. 7.

3 **Queuing:** If a merged segment was created, add the new segment to the queue and dequeue the merged segments.

5 Examples

In this section, we compare MLSD to LSD using images that specifically illustrate the limitations of LSD. Computation time are given for both methods.

5.1 Low-contrast images

We first consider an image with low contrast (Fig. 7). As can be seen when zooming the picture (see Fig. 8), the image is also noisy. As the gradient is also noisy around edges, the tile borders are hardly detected at all, whereas MLSD does detect them and does not create much over-segmentation.

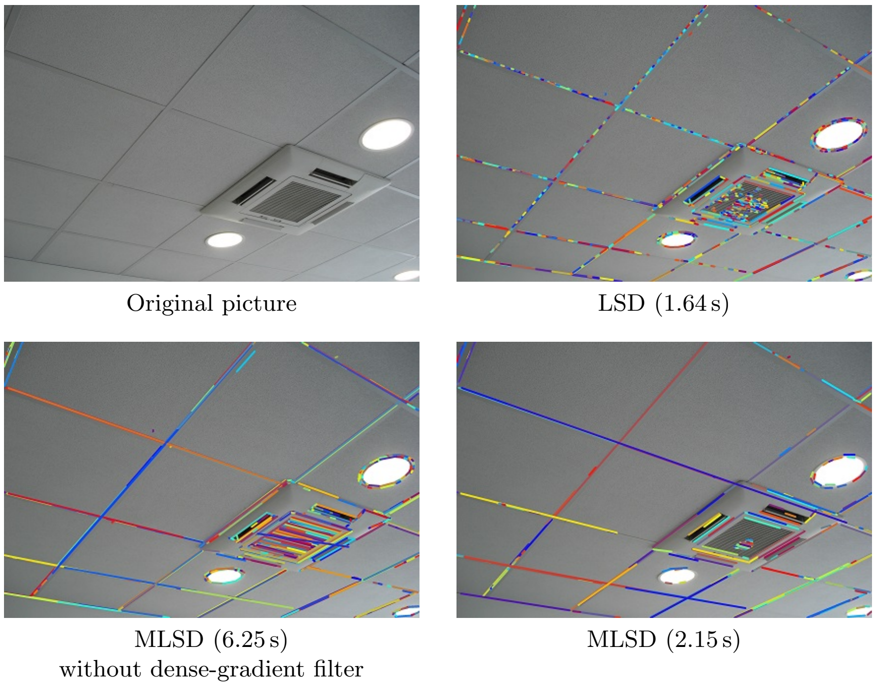


Fig. 9: Comparison between MLSD with and without dense-gradient filter.

5.2 Dense-gradient filter

Fig. 9 illustrates the efficiency of the dense-gradient filter. The main difference between the two MLSD results occurs in the central part of the image where there is a grid pattern whereas the other parts are not affected. This type of pattern tends to slow the algorithm a lot (typically by a factor 3). Moreover, as argued above, the added segments are similar to each other and tend to deteriorate matching, and thus SfM results.

5.3 Effect on scaling and crop

Fig. 10 and 11 illustrate the differences between LSD and MLSD for images with different resolutions. In Fig. 10, we decreased the size of the image (four times in both height and width) and in Fig. 11, we increased the size of the image (four times in both height and width).

Fig. 12 illustrates the differences between LSD and MLSD for cropped versions of the same image. We cropped the original picture into a 2 times smaller picture and then in a 4 times smaller picture. Whereas MLSD does not show significant changes from one picture to the other, LSD tends to detect differently segments.

Although in each case the results are different for both algorithms, the changes are limited for MLSD, whereas LSD gives sensibly different results with either a change of resolution or a crop.

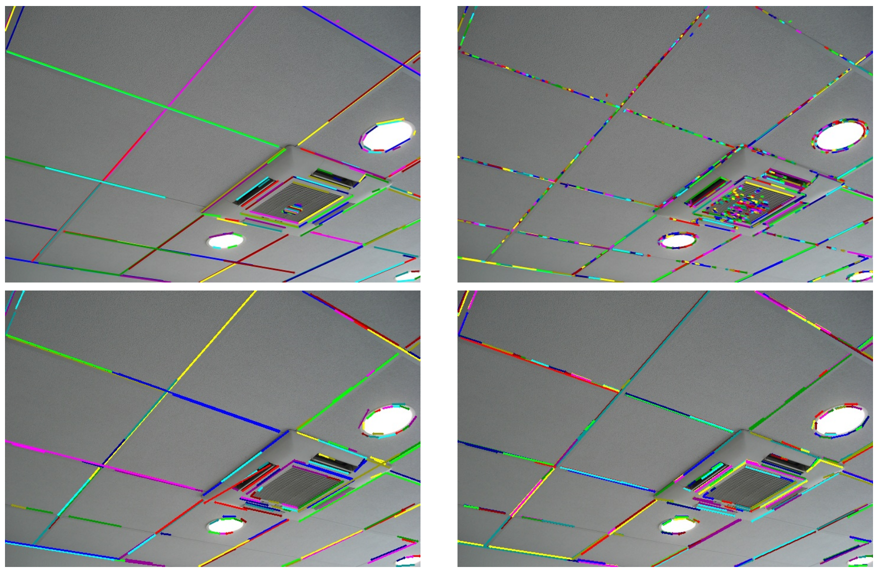


Fig. 10: Comparison between LSD and MLSD on the same image with original resolution (above, 5 Mpixels) and four times reduced (below, 330 kpixels).

6 Conclusion

We presented MLSD, a multiscale extension to the popular Line Segment Detector (LSD). MLSD is less prone to over-segmentation and is more robust to noise and low contrast. Being based on the a contrario theory, it retains the parameterless advantage of LSD, at a moderate additional computation cost. The source code accompanying this paper is not yet at as clean and readable as it could be. We plan to clean it and build an online demonstration with it on the IPOL website (`http://www.ipol.im/`).

References

1. C. Akinlar and C. Topal. EDLines: A real-time line segment detector with a false detection control. *Pattern Recogn. Lett.*, 32(13):1633–1642, Oct. 2011.

2. R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a line segment detector. *Image Process. On Line (IPOL 2012)*, 2:35–55, 2012. `http://dx.doi.org/10.5201/ipol.2012.gjmr-lsd`.
3. Y. Salaün, R. Marlet, and P. Monasse. Multiscale line segment detector for robust and accurate SfM. In *23rd International Conference on Pattern Recognition (ICPR)*, 2016.
4. Y. Salaün, R. Marlet, and P. Monasse. Robust and accurate line- and/or point-based pose estimation without Manhattan assumptions. In *European Conference on Computer Vision (ECCV)*, 2016.

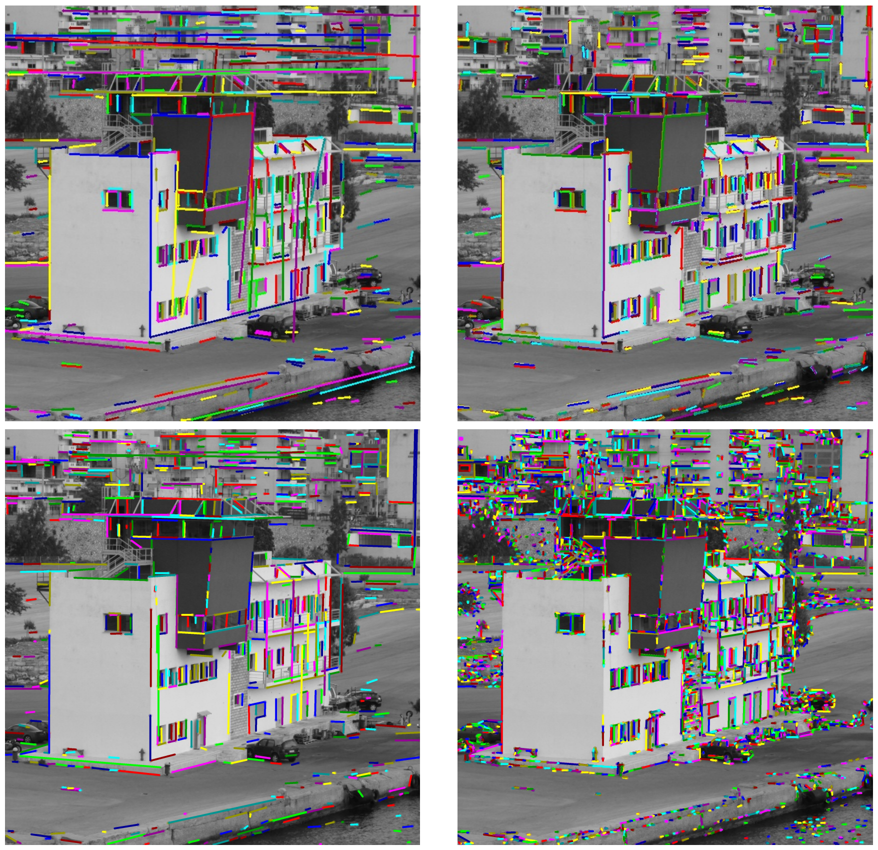


Fig. 11: Comparison between LSD and MLSD on the same image with original resolution (above, 360 kpixels) and resolution four times larger (below, 5.8 Mpixels).

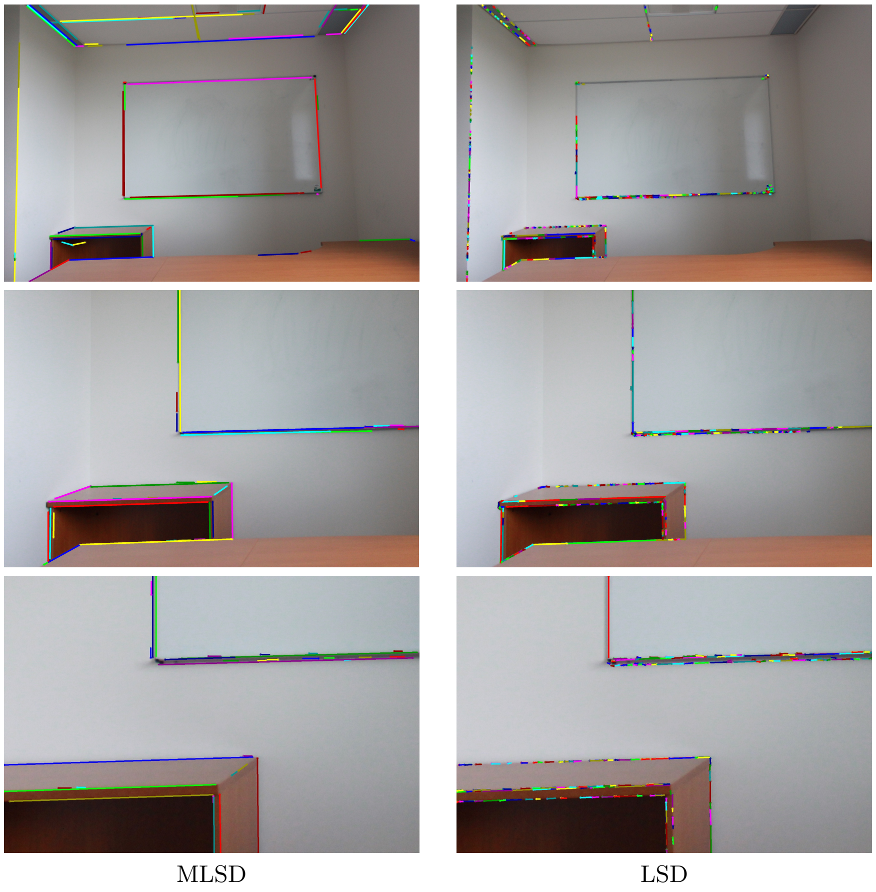

MLSD LSD

Fig. 12: Comparison between LSD and MLSD on the same image (top, 18 Mpixels) but with cropped versions (no resolution changes) of respectively 5.6 Mpixels (middle) and 1.5 Mpixels (bottom).