# MSER: Implementation and Comparison with the OpenCV Version

Computational Vision course - A.Y. 2024/2025

Sara Caviglia - 5163676

January 2026

## Abstract

The use of MSER has spread in various contexts since the publication of the first paper. We propose a custom implementation of the algorithm, following the description proposed by Matas et al. in 2002 [5]. Since the implementation in OpenCV follows a different approach, we compare the results in terms of time, regions, and similarity. To conclude, we open the possibility of using MSER in the context of licence plates detection.

## 1 Introduction

With MSER we indicate *Maximally Stable Extremal Regions*. They are a particular kind of extremal regions, which are a subset of distinguished regions. Extremal regions have two desirable properties: the set is closed under continuous one-to-one (and thus perspective) transformation of image coordinates and, secondly, it is closed under monotonic transformation of image intensities [5].
Since they are features, they can be used for multiple tasks in Computer Vision. The original paper describes the algorithm to find MSER, as well as the method to use them in the estimation of epipolar geometry. The context of robust wide-baseline stereo is a preferred application because MSER are features that carry shape information.

## 2 State-of-the-Art, Background, and Related Works

The paper was published in 2002, but the algorithm is still cited in many articles, across different subjects [7]. The authors referenced various stereo matching and couple recognition algorithms with a common structure: the methods used local invariant descriptors.
In the list of related works, we find the Scale Invariant Feature Transform (SIFT) [4].

## 3 Algorithms and Methods

### 3.1 General description of the algorithm

The algorithm proposed is the following.

1. The pixels of the image are sorted by intensity. The algorithm suggested is `BINSORT`, which has linear complexity ($\mathcal{O}(n)$).
2. The pixels are placed in the image in order (either increasing or decreasing; for this implementation we used increasing order). The list of connected components and their areas is maintained using the union-find algorithm.
3. We now have a data structure that stores the area of each connected component as a function of intensity.
4. Intensity level that are local minima of the rate of change of the area function are selected as thresholds, producing MSER.

## 3.2 Implementation of the algorithm

The implementation of the algorithm and the code used for testing are available in a GitHub repository [2].
The core of the method is the class `myMSER`, which includes the main function `enumMSER`, the function for non-maximum suppression (NMS) `nmsMSER`, and other auxiliary functions. In this class, we reference the class `UnionFindDS`, which implements the data structure, and the function `binsort`, previously defined.
The parameters used in the custom implementation are the same as provided by OpenCV [1].

- `delta` is used to compare intensities in pixels at a certain distance.
- `min_area` is the minimum area that a connected component has to be in order to be considered a region.
- `max_area` is the maximum area that a connected component can be in order to be considered a region.
- `max_variation` is the maximum variation of intensity I can have within the same region.
- `min_diversity` is the minimum diversity I can have within the same region.

The other implemented functions are used to test the custom implementation, to make comparisons with the OpenCV implementation, and to visualise the regions with different methods.

## 4 Experiments

To test the implementation, especially towards a licence plate recognition algorithm, we used a specific image (Figure 1), taken from a wider dataset [3]. The tests were carried out also with other images, which were not suited for continuous testing since the computation time was higher.
We can define and separate three different tests that were carried out.

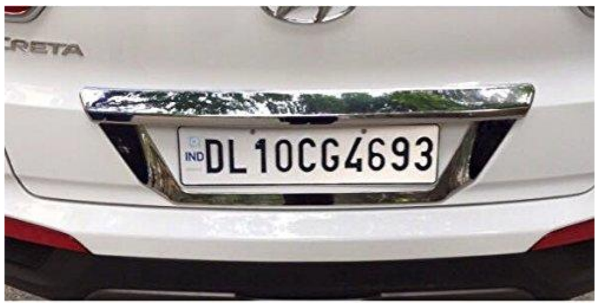

Figure 1: Test image used

1. `test1`: Comparison between the one-pass custom approach and two-pass custom approach
2. `test2`: Comparison between the two-pass custom approach and the OpenCV implementation
3. `test3`: Comparison between the two-pass custom approach and the OpenCV implementation, both with `delta` = 10, instead of the default value

For every case, we also computed the time needed by each approach, the number of regions identified, and the intersection over union (IoU) between the regions identified by the two approaches considered.

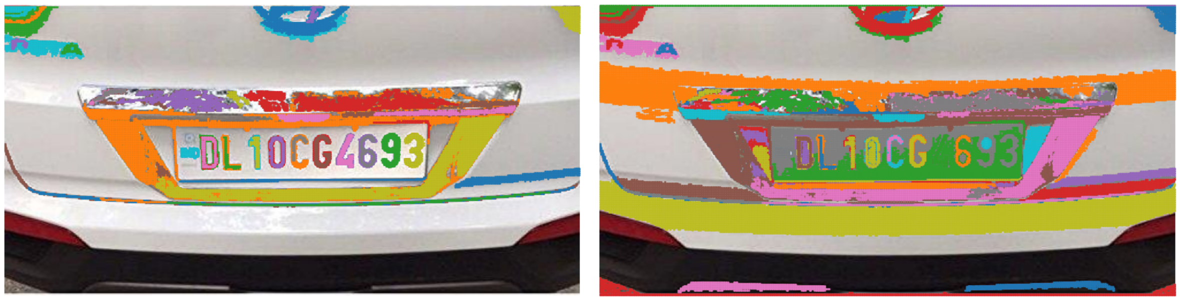

Figure 2: Regions found with a one-pass approach (on the left), and regions found with a two-pass approach (on the right).

### 4.1 `test1`

OpenCV uses a two-pass approach, where the enumeration of the MSER is firstly computed on the image, then on the inverted image; lastly, the regions are united. The Figure 2 clearly shows that, with the two-pass approach, the regions found cover more of the image. As depicted in Table 1, we can see that the number of regions found increases with the second approach, as well as the time needed to compute it. However, the IoU between the two sets of regions is only 0.3684.

| | Time (s) | Number of regions |
|---|---|---|
| One-pass | 1.5145 | 105 |
| Two-pass | 2.9954 | 173 |

Table 1: Metrics for `test1`

### 4.2 `test2`

The second test aimed to find the differences between the custom implementation and the OpenCV one. We can look at the representation in Figure 3 to get a grasp of the difference. It is clear that the custom implementation is rougher and less detailed. The biggest difference can be found in the licence plate: while the OpenCV implementation recognises the whole, it does not happen in the custom. The metrics of this test clearly show that the custom implementation is much slower, and finds less regions (Table 2). However, the IoU is 0.7596, indicating a medium-high similarity between the found regions.

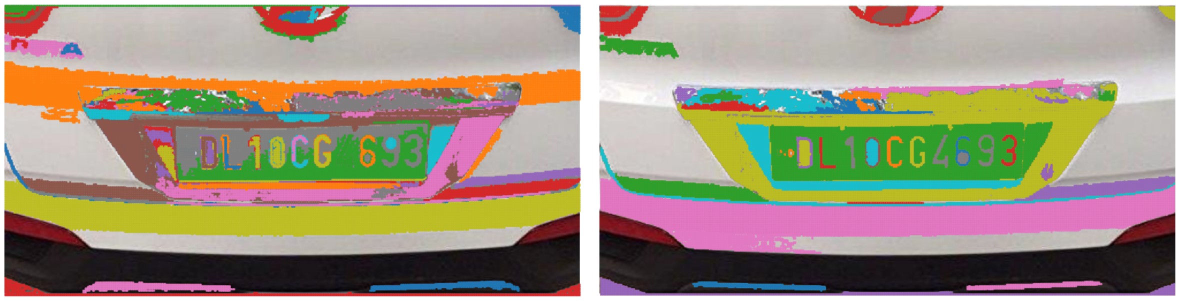

Figure 3: Regions found with custom implementation (on the left), and regions found with OpenCV implementation (on the right).

### 4.3 `test3`

With the last test carried out, we aimed to modify the `delta` parameter, which defines the grain of the thresholding [1]. Since we *increased* `delta`, we obtained a finer-grained thresholding. With this new

|  | Time (s) | Number of regions |
|---|---|---|
| Custom | 3.0376 | 173 |
| OpenCV | 0.0113 | 620 |

Table 2: Metrics for `test2`

value, the regions found, seen in Figure 4, are more similar than the ones in the previous tests. The computing time is aligned with the time needed in `test2`, but the number of regions changes (Table 3): this difference is bigger for the OpenCV implementation than for the custom one. We can assume that OpenCV finds a lot of small regions that with a higher `delta` are not found. In this case, the IoU is the highest, with a value of 0.8650.

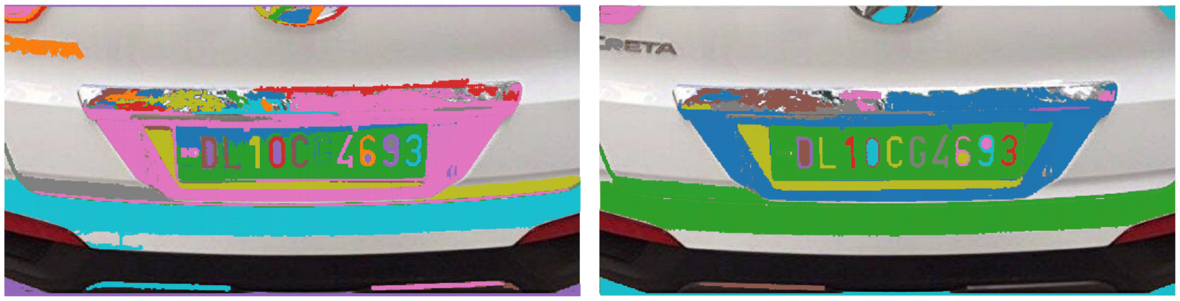

Figure 4: Regions found with custom implementation (on the left), and regions found with OpenCV implementation (on the right).

|  | Time (s) | Number of regions |
|---|---|---|
| Custom | 3.4027 | 127 |
| OpenCV | 0.0096 | 370 |

Table 3: Metrics for `test3`

## 5 Conclusion and Future Works

### 5.1 Considerations on the tests

Referencing the three tests seen, we can say that this custom implementation is slower than the one implemented by OpenCV, and it finds less regions. We can see two reasons behind this consideration.

- The custom implementation uses non-maximum suppression (NMS) and diversity, while OpenCV uses pruning. These are both approaches with the goal of avoiding duplicates, but they work on different concepts. NMS is a spatial algorithm, which verifies if two elements, in this case two regions, are in the same spot. On the other hand, pruning is hierarchical, so it looks if one element is *inside* another one. In the context of MSER, pruning works directly on the union-find structure.

- There is a difference in the computational ordering of the pixel. While the custom implementation uses connected components, as suggested in [5], the OpenCV implementation uses a computational ordering suggested by an immersion analogy and presented in [6]. This method allows the implementation of the algorithm in quasi-linear time, while the custom implementation is obviously more costly.

These considerations about the results are the same for more complex examples: in images with more regions, OpenCV performs better. A test with a different image can be found in the repository [2], in the notebook `CavigliaSara_other.ipynb`.

## 5.2 Towards licence plate detection

The detection of MSER can be used to detect text, like licence plates. For this reason, we provide a visualisation with bounding boxes. In Figure 5 we can see the bounding boxes obtained with the functions and parameters of `test2`. However, it is clear that the boxes are still too much, and it seems impossible to find the ones containing the licence plate. The majority of European licence plates follow one of the basic standard worldwide [8], which dictates that they have to be 520 by 110 millimetres, or 520 by 120 millimetres. We used this ratio to visualise bounding boxes that are candidate licence plates (Figure 6). This can be the first step of licence plate detection, followed by convolutional neural networks (CNNs) or optimal character recognition (OCR).

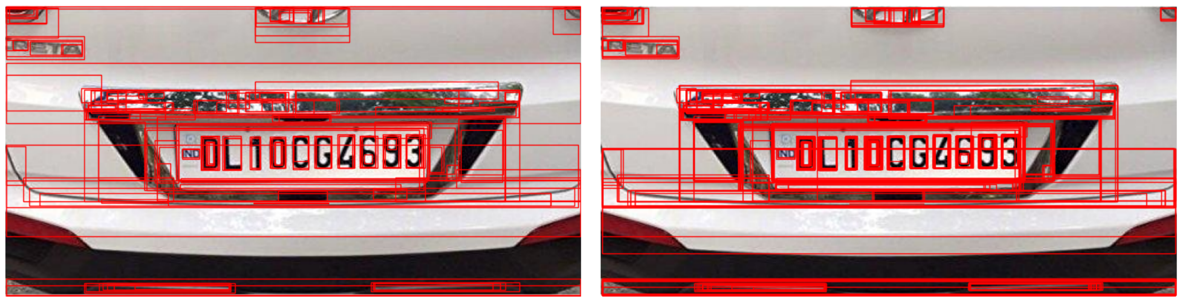

Figure 5: Bounding boxes found with custom implementation (on the left), and bounding boxes found with OpenCV implementation (on the right).

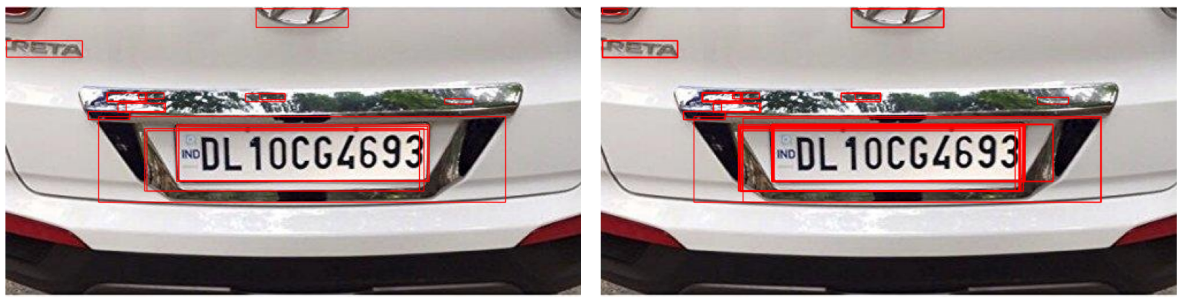

Figure 6: Candidate licence plates found with custom implementation (on the left), and candidate licence plates found with OpenCV implementation (on the right).

## References

[1] Subramanya G Bellary. Mser (maximally stable extremal regions). [Online; accessed 20-January-2026]. URL: https://medium.com/@roronoa-zoro/mser-maximally-stable-extremal-regions-93ae022bccf6.

[2] Sara Caviglia. Cv-project - github repository. URL: https://github.com/saracaviglia/CV-project/.

[3] Unique Data. License plates - 2,6m+ plates, ocr dataset. URL: https://www.kaggle.com/datasets/trainingdatapro/license-plates-1-209-438-ocr-plates/data.

[4] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi:10.1109/ICCV.1999.790410.

[5] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004. British Machine Vision Computing 2002. URL: https://www.sciencedirect.com/science/article/pii/S0262885604000435, doi:10.1016/j.imavis.2004.02.006.

[6] David Nistér and Henrik Stewénius. Linear time maximally stable extremal regions. volume 5303, pages 183–196, 10 2008. doi:10.1007/978-3-540-88688-4_14.

[7] Scopus. Document search result - matas et al. [Online; accessed 19-January-2026]. URL: https://www.scopus.com/results/results.uri?cc=10&sort=plf-f&src=s&nlo=&nlr=&nls=&imp=t&sid=400e99ae7e3eaa603ec765ac4afc1349&sot=cite&sdt=a&sl=22&s=REF%282-s2.0-3142736062%29&ss=plf-t&ps=r-f&editSaveSearch=&origin=resultslist&zone=resultslist&sessionSearchId=400e99ae7e3eaa603ec765ac4afc1349&limit=200.

[8] Wikipedia contributors. European vehicle registration plate — Wikipedia, the free encyclopedia, 2025. [Online; accessed 22-January-2026]. URL: https://en.wikipedia.org/w/index.php?title=European_vehicle_registration_plate&oldid=1329425920.