applied sciences

MDPI

Article

Efficient Edge Detection Method for Focused Images

Agnieszka Lisowska

Institute of Computer Science, University of Silesia, ul. Bedzinska 39, 41-200 Sosnowiec, Poland; agnieszka.lisowska@us.edu.pl

Abstract: In many areas of image processing, we deal with focused images. Indeed, the most important object is focused and the background is smooth. Finding edges in such images is difficult, since state-of-the-art edge detection methods assume that edges should be sharp. In this way, smooth edges are not detected. Therefore, these methods can detect the main object edges that skip the background. However, we are often also interested in detecting the background as well. Therefore, in this paper, we propose an edge detection method that can efficiently detect the edges of both a focused object and a smooth background alike. The proposed method is based on the local use of the k-Means algorithm from Machine Learning (ML). The local use is introduced by the proposed enhanced image filtering. The k-Means algorithm is applied within a sliding window in such a way that, as a result of filtering, we obtain a given square image area instead of just a simple pixel like in classical filtering. The results of the proposed edge detection method were compared with the best represented methods of different approaches of edge detection like pointwise, geometrical, and ML-based ones.

Keywords: edge detection; local k-Means; sliding window

Citation: Lisowska, A. Efficient Edge Detection Method for Focused Images. *Appl. Sci.* **2022**, *12*, 11668. https://doi.org/10.3390/app122211668

Academic Editors: Andrés Márquez and Zhengjun Liu

Received: 7 October 2022
Accepted: 11 November 2022
Published: 17 November 2022

1. Introduction

Edge detection methods are very commonly used in many areas of image and video processing and computer vision. They are commonly used in object detection, classification, and recognition. This is because edges are the most important objects, from the Human Visual System point of view [1], that are present in images. There are many approaches to edge detection. Starting from the simple image filtering methods, which are extremely fast, through more advanced geometrical methods, which are used in shape representation, up to machine learning-based methods, especially deep convolutional neural networks, which need prior data training [2].

The main problem of efficient edge detection follows from the fact that all edge detection methods work with a global assumption that the edge is defined as a step edge—a sharp change of image intensity. In fact, in the case of still images, we deal with edges of different levels of sharpness and smoothness. The most representative case is an image that is focused—it has a sharp foreground and smooth background. So, treating all edges in the same global way must be inefficient during edge detection of such an image. Thus, to overcome this problem, a method that works locally has to be applied, according to the known and very deep thought "Think globally, act locally". By working locally, such a method could adaptively adjust the definition of an edge depending on the local background. It means that in the sharp region, this method can detect sharp edges, while in smooth regions, it can detect smooth edges.

In this paper, we address the problem of edge detection in focused images. In such a case, it is difficult to detect both kinds of edges simultaneously. The existing methods either do not detect smooth edges in the background or detect too much noise in the foreground. In this paper, we propose an approach that can detect all edges in focused images efficiently. The efficiency follows from the fact that the proposed method works locally and adaptively detects edges inside a sliding window. This method is based on the k-Means algorithm

well known in Machine Learning. The use of the k-Means algorithm locally gives a good balance between the speed of the pointwise methods and the efficiency of the CNN-based methods. To use the k-Means algorithm locally, we proposed the enhanced version of image filtering—we filter an image not pixel-by-pixel, but area-by-area. The proposed algorithm of edge detection was compared to the state-of-the-art methods taken from different classes of edge detection methods.

This paper is organized in the following way. In Section 2, the related work is presented. In Section 3, the problem statement is posed. In Section 4, the new method is proposed. In Section 5, the experimental results of edge detection are presented. Finally, Section 6 concludes this paper.

2. Related Work

Edge detection methods have a long history. The most commonly known class of edge detection methods is based on the pointwise approach. The main known icons are the Roberts algorithm [3], Prewitt operator [4,5], Sobel method [6,7], and the most commonly used Canny algorithm [8]. These methods are mainly defined as filters based usually on the first or the second derivative of the function representing an image. These derivatives are often combined with the pre- or postprocessing methods or are improved in different ways [9–13]. These algorithms are fast but are not noise resistant.

Another class of edge detection methods is the one represented by the geometrical approach. The most known is the method based on the Hough transform [14]. Another approach is based on moment computation [15,16]. More recently, a new method was developed based on the multiresolution geometrical edge detection [17–19]. All the geometrical methods treat edges as line or curve segments. From this follows that they are rather slow in comparison to the pointwise methods, but they treat shapes in a geometrical way and are noise resistant. Thanks to that, these methods can be used as feature extractors in object recognition.

A quite different class of methods is based on the Machine Learning approach. The most known algorithms are the random forests [20–22] and the k-Means algorithm [23–25]. There are also different variations of such methods [26,27]. These methods are efficient and are relatively not time-consuming. They are more sophisticated than pure filtering methods since they analyze the image content.

Finally, these days, many algorithms are based on Deep Learning of convolutional neural networks (CNN). The first attempt to change the classical approach from bottom-up to top-down multiscale edge detection was proposed with the use of DeepEdge [28]. Another known algorithm is HED [29], which is also multiscale. Then a number of different approaches were proposed based on CNN [30–34]. Recently, a simple, lightweight, and efficient algorithm was proposed based on the Pixel Difference Network [35]. All CNN methods are based on feature detection. Thus, they can detect edges in a more intelligent way than simple pointwise algorithms. However, these methods require a priori a huge amount of learning data to be further trained and used. Additionally, the training is time-consuming.

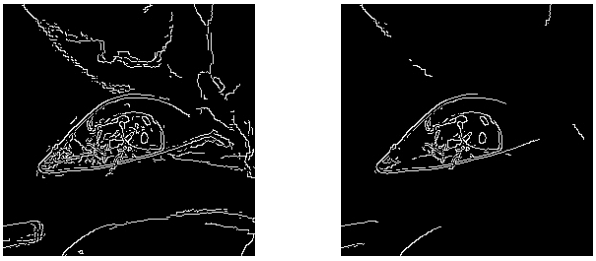
3. Edge Detection

The efficient state-of-the-art edge detection methods assume that one deals with well-defined edges in an image and try to detect them. However, sometimes we deal with images that are highly focused, like the example presented in Figure 1. In such images, we are still interested in the detection of both the foreground object and the background smooth edges, as the focus was made artificially by a photographer, usually, for artistic reasons. According to our knowledge, there are no efficient methods that can deal with sharp and smooth edges at the same time.

Figure 1. The example of a focused image.

3.1. Limitations of the Existing Models

We can observe the above-mentioned problem in more detail in Figure 2. We used the well-known Canny edge detector [8] for edge detection in the sample image from Figure 1. In Figure 2a, we can see the edges detected in such a way as to catch the smooth edges together with the sharp ones (the low threshold was fixed as 50 and the high threshold as 100 to catch the smooth edges). However, we can see the noise in the foreground as well. In fact, we do not want to see it. On the other hand, we can see another example of edge detection in Figure 2b. This time, the thresholds were fixed to remove the noise from the foreground (the low threshold was fixed at 150 and the high threshold at 200 to remove the noise). However, some of the background edges were removed as well. This holds for all edge detectors proposed so far. So, in fact, we are looking for a compromise: detect edges without noise in the foreground and to preserve correctly the background edges. To do this, we propose an algorithm that is based on the local k-Means method from Machine Learning.



(**a**) lowTh = 50, highTh = 100 (**b**) lowTh = 150, highTh = 200

Figure 2. Edge detection by the Canny method for the low and high thresholds that are equal to: (**a**) 50 and 100 (**b**) 150 and 200, respectively.

3.2. k-Means Algorithm

The k-Means algorithm [36] is the Machine Learning method that is used for partitioning n observations into k clusters in such a way that each observation belongs to the cluster with the nearest mean. This method is also applied in image segmentation. An image is segmented into k segments assuring the smallest Mean Square Error (MSE). The examples of k-Means segmentation are presented in Figure 3 for different numbers of segments k together with the edges between segments.

One can observe from Figure 3 that the k-Means algorithm can detect edges of any smoothness, since it is based on image segmentation and not on edge detection. However, to get satisfactory results, one first has to fix the number of segments first. However, direct application of the k-Means algorithm for edge detection in focused images is not efficient since the same drawback is presented as in the case of e.g., Canny's algorithm. Indeed, the more segments we fix, the better detection of background edges we obtain, but the more noise in the foreground appears as well (see image Figure 3d).

(**a**) segments, $k = 3$ (**b**) segments, $k = 8$ (**c**) edges, $k = 3$ (**d**) edges, $k = 8$

Figure 3. Edge detection by the k-Means algorithm. The upper row: k-Means segmentation for (**a**) $k = 3$, (**b**) $k = 8$ segments. The lower row: the edges of k-Means segmentation for (**c**) $k = 3$, (**d**) $k = 8$ segments.

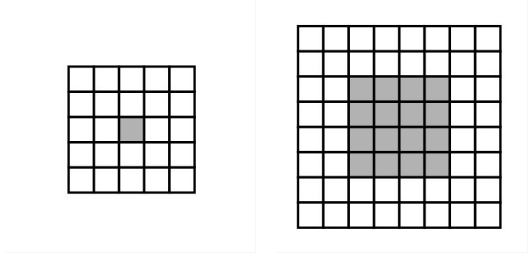
4. Edge Detection by the Local k-Means

In this section, we propose an efficient method for edge detection in focused images, which is based on the proposed use of the k-Means algorithm locally. Though the idea of local use of the k-Means algorithm through the sliding window was already proposed [37], it was performed quite differently and was dedicated to specific tasks like text document analysis.

As we could see in the previous section, the application of the k-Means algorithm to the whole image does not lead to satisfactory results of edge detection. Therefore, we propose to apply k-Means locally via a filtering window, which goes through the image, like a typical filter, and compute k-Means in this local window. However, unlike in the classical filtering process, we compute here a subimage within the filter instead of a single pixel. It means that, unlike in the state-of-the-art methods, we obtain as a result a subimage instead of a pixel.

In more detail, to compute the convolution of an image with the typical filter of size 5×5 pixels, see Figure 4a, we apply the filtered pixel by pixel to the image (in the horizontal and then the vertical direction) and compute the new pixel's value each time taking into account the values within this 5×5 pixels window. However, in our case, we apply the filter to the given square area, see Figure 4b. In other words, we divide the given image into subimages of size e.g., 4×4 pixels and for each subimage we apply the area mask (of size, e.g., 8×8 pixels) and compute the k-Means algorithm, within this 8×8 pixels window. Then we draw the result within just the 4×4 pixels area, which is the considered subimage. Similarly, as in classical filtering, in the proposed method, we deal with edge pixels that go beyond the image during filtering. We solve this problem by reducing the mask's size to the area size in border squares.

The proposed method is summarized in Algorithm 1. We fix the initial values of the segments and means on lines 1–2. In lines 4–12, the classical k-Means algorithm is presented. We iterate the segmentation according to the means (lines 6–7). In each step, the means are updated (lines 8–9). The algorithm stops when the assumed error measured as the Mean Square Error (MSE) between the original image and the segmented one (line 11) is small enough. Next, the local application of this algorithm is defined in lines 13–17. The image is divided into adjacent subsquares of size *areaSize* (lines 13–14). Then, for each such subsquare, the accompanying mask is defined of size *maskSize* (line 15). Next, the k-Means algorithm is computed for such an image within the mask (line 16). Finally, just the small subimage of size *areaSize* is drawn as segmented (line 17). Note that, depending on an image, we can adjust the sizes of the filtering window and the considered area. Note that when we fix these sizes as the same, we deal with a simple application of the k-Means algorithm in squared subimages of the considered image.



(**a**) point mask 5×5 (**b**) area mask 8×8

Figure 4. The sample filter masks: (**a**) a point mask of size 5×5 pixels, used for i.e., Canny algorithm, (**b**) an area mask of size 8×8 pixels, with the area size of 4×4 pixels, used for the proposed local k-Means.

Algorithm 1: The local k-Means edge detection.

Input: *img* – image; k – number of segments; *areaSize* – size of the considered area; *maskSize* – size of the filtering mask.
Output: An image with detected edges.

fix k, *areaSize*, *maskSize*
fix $w_i^{(0)}, i \in \{1, \ldots, k\}$
t = 0

function kMeans(*img*):
while $error < \epsilon$ **do**
  for $i = 1$ *to* k **do**
    $V_i = \{img(x,y) : |img(x,y) - w_i^{(t)}| \leq |img(x,y) - w_j^{(t)}|, j \in \{1, \ldots, k\}\}$
  for $i = 1$ *to* k **do**
    $w_i^{(t+1)}$ = mean of all elements of V_i
  t=t+1
  $error = MSE(img, allSegmentsV_i)$
return segmented *img*

for $i = 0$ *step areaSize to* n **do**
  for $j = 0$ *step areaSize to* m **do**
    $subImg(maskSize) = img(i, j, maskSize)$
    kMeans($subImg(maskSize)$)
    draw $subImg(areaSize)$

5. Experimental Results

In this section, we present the experimental results of edge detection. In Figure 5, the tested benchmark images [38] are presented. These images were resized to 256×256 pixels to make the computations easier. However, the proposed algorithm can be applied to images of any size.

To fix the optimal parameters used in the proposed method (i.e., the number of segments, the size of the area window, and the size of the mask window), a number of experiments were performed. We show them for a sample image "Ladybird".

The edges detected by the proposed local k-Means algorithm for different sizes of the area window (i.e., subimage) are presented in Figure 6. In this example, the number of segments were fixed at $k = 3$. From these images, one can see that the best visual results one obtains for the size equal to 32×32 pixels. Smaller areas (especially 8×8) cause the noise effect. So, for all our experiments, we fixed the area size as 32×32 pixels.

(**a**) Ladybird (**b**) Monarch (**c**) Parrots

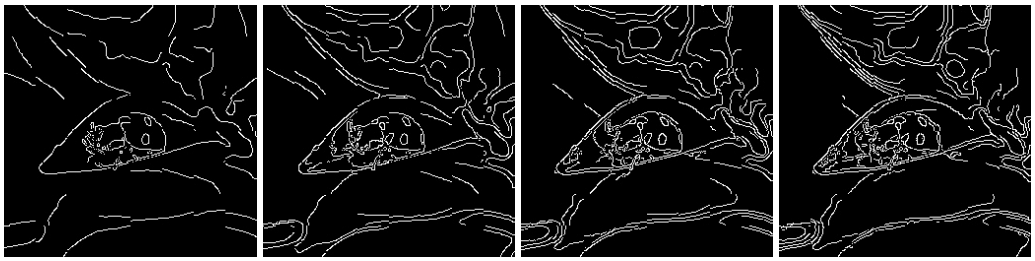
(**d**) Bee (**e**) Soup (**f**) Balls

Figure 5. The benchmark images used in the experiments.

The edges detected by the proposed algorithm for different numbers of segments k are shown in Figure 7. From these images, one can see that the best results are obtained for three or four segments. In further experiments, we use three segments.

(**a**) size = 8×8 (**b**) size = 16×16 (**c**) size = 32×32 (**d**) size = 64×64

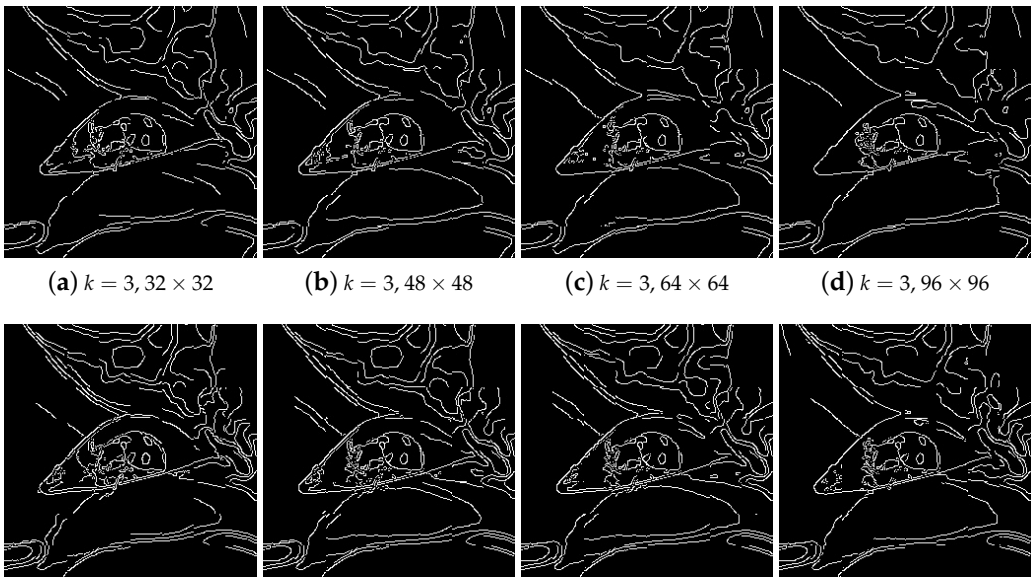
Figure 6. Edges detected by the local k-Means algorithm with different area (subimage) size.



(**a**) $k = 2$ (**b**) $k = 3$ (**c**) $k = 4$ (**d**) $k = 5$

Figure 7. Edges detected by the local k-Means algorithm with different numbers of segments.

The final test was made to check the optimal size of the mask. Therefore, the edges detected by the proposed algorithm for different sizes of the masks are shown in Figure 8. The size of the area was fixed as 32×32 pixels. From these images, one can observe that the optimal results are obtained for the mask's size 48×48 or 64×64 pixels.



(**a**) $k = 3, 32 \times 32$ (**b**) $k = 3, 48 \times 48$ (**c**) $k = 3, 64 \times 64$ (**d**) $k = 3, 96 \times 96$

(**e**) $k = 4, 32 \times 32$ (**f**) $k = 4, 48 \times 48$ (**g**) $k = 4, 64 \times 64$ (**h**) $k = 4, 96 \times 96$

Figure 8. Edges detected by the local k-Means algorithm with a different number of segments (rows: $k = 3$, $k = 4$) and different sizes of the mask (columns: 32×32, 48×48, 64×64, and 96×96 pixels).

From the experiments performed, one can conclude that the optimal size of the subimage and the optimal number of segments can be fixed globally for all tested images. However, in the case of the mask's size, we can observe that, depending on the image, the best results are obtained by the sizes oscillating somewhere between 48×48 and 64×64 pixels. However, fixing this parameter for all images give also good results.

Usually, when we deal with edge detection methods, the noise resistance is tested. However, in our case, it can be skipped. The reasons are twofold. Firstly, we are interested in focused images. This kind of image is noise-free by definition. Secondly, even if we would like to consider noised images, there is a number of methods to remove noise from k-Means clustering efficiently, e.g., [39], that can be applied.

Finally, we compared the proposed method to the state-of-the-art ones. The reference methods are Canny, wedgelets2 [19], and global k-Means. The Canny and k-Means algorithms are classical and work globally and pointwise (however, the latter method works in a more intelligent way than the former one). However, the wedgelets2-based method was proposed as the geometrical method that can be used for object detection or recognition. This method is local and is based on the local window mechanism. These reference algorithms were chosen as the best methods representing different approaches (i.e., pointwise, geometrical, and ML-based). We excluded CNN-based methods from the experiments since they need a huge training dataset and time-consuming training.

In all the methods tested, the optimal parameters were fixed. In the Canny case, the thresholds were fixed as 100 and 150 as this is the best compromise between the lack of noise in the foreground and the accuracy of the background edges. In the case of wedgelets2, the second-order wedgelets were used. This method works in a geometrical way, so it can better detect background edges than the Canny method. In the case of the k-Means algorithm, the number of segments was used as 8 to find the compromise between the accuracy of the background and the foreground edges detection.

To show the advantages of the proposed method we first show a simple artificial example from Figure 5f. In this example, one of the balls is in motion, so it is not focused. In Figure 9 the results of edge detection for this image by different methods are presented. From these results we can see that: (1) the Canny method cannot detect the object in motion properly (the edges are too smooth to be caught by this method); (2) the wedgelets2-based method produces the edges in a manner of small lines or curves, not necessarily connected;

(3) the k-Means algorithm detects the shadows on the table and this cannot be avoided without significant degradation of the balls, it follows from its global working; (4) the proposed method seems to overcome all the drawbacks of the former methods. Indeed, it can detect smooth edges as well as sharp ones, it produces nearly continuous edges, and thanks to the locality, it avoids detecting unexisting edges like shadows.

(**a**) Canny (**b**) wedgelets2 (**c**) k-Means (**d**) proposed

Figure 9. The results of edge detection by different methods: (**a**) Canny, (**b**) wedgelets2, (**c**) k-Means, (**d**) local k-Means (proposed).

In Figure 10 the results of edge detection by different methods for natural still images are shown. From the presented results, one can see that the proposed local k-Means algorithm definitely works better than the reference algorithms. For this method, all edges are just edges, no matter how smooth they are.

In the above comparisons, we have limited to the visual assessment of the edge detection accuracy. It follows from the fact that we compared here the methods from different classes of edge detection techniques. Each class has specific characteristics and needs a different approach in the objective evaluation of edge detection efficiency. This is because of the different edge definitions used. Let us note that in the case of pointwise methods, a detected edge is a pure set of points, whereas, in the case of geometrical methods, a detected edge is a segment of a line or curve. The second point is that when we deal with sharp edges, it is relatively easy to decide what is an edge and what is not. When introducing smooth edges, it is hard to clearly state whether we still deal with an edge or a texture, or something else. It depends on the application and the user's needs. However, to show that the introduction of the local k-Means really improves edge detection numerically, we present here the comparison of the image segmentation results between the original global k-Means algorithm and the proposed local use of it. As one can see from Table 1, the proposed method gives better-quality image segmentation in the Peak-Signal-to-Noise-Ratio (PSNR) sense. So, we can conclude that since the proposed method segments images more accurately than the k-Means algorithm, it also better represents the detected edges than the original method.

Table 1. Image segmentation quality for global (original) and local (proposed) use of the k-Means algorithm in the means of PSNR.

PSNR	Global k-Means	Local k-Means
Ladybird	30.26	31.68
Monarch	29.38	30.16
Parrots	29.13	30.81
Bee	29.13	30.2
Soup	28.84	30.12
Balls	31.40	35.51

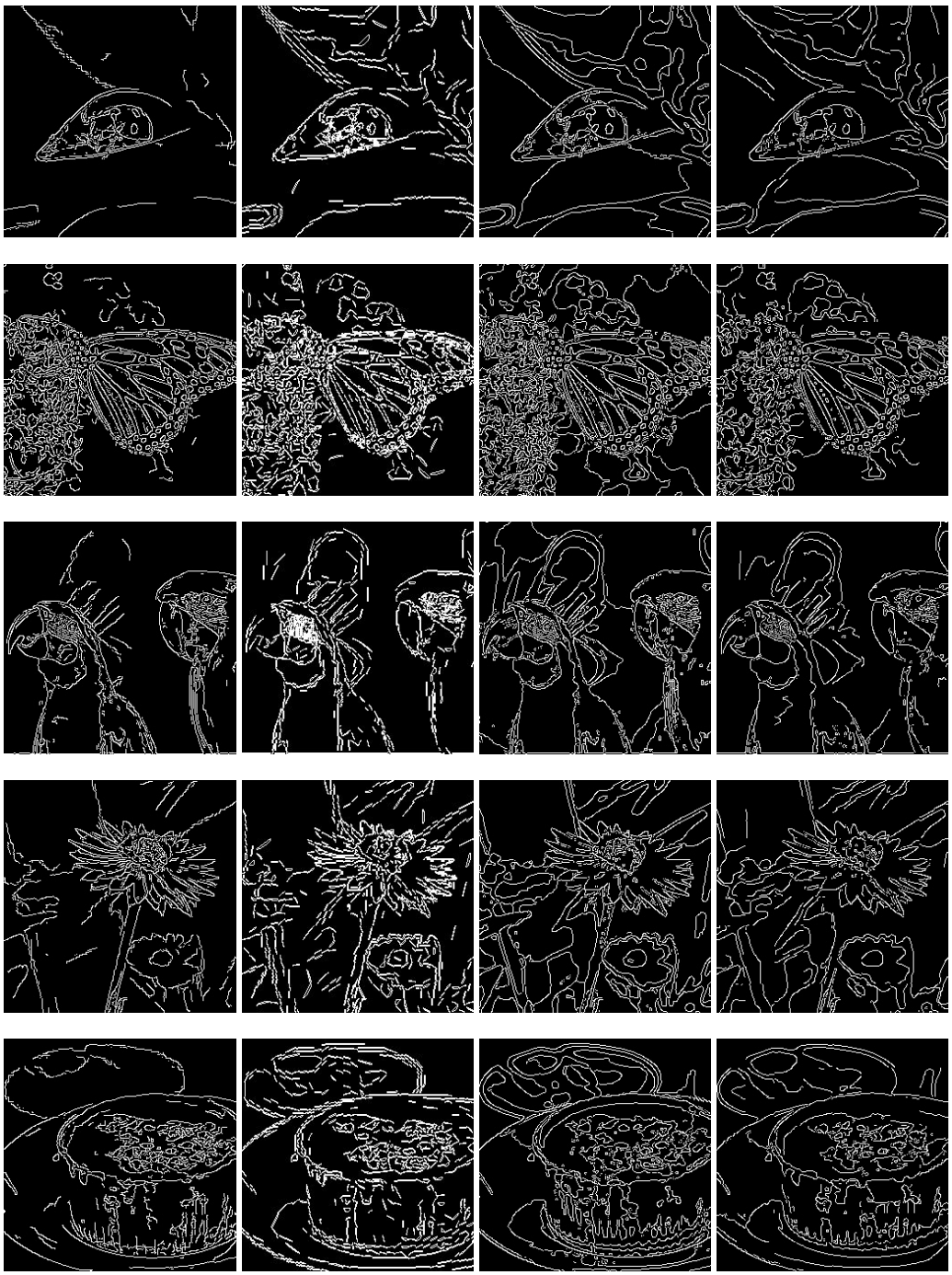


Figure 10. The results of edge detection by different methods: first column—Canny, second column—wedgelets2, third column—k-Means, fourth column—local k-Means (proposed).

Let us note that the time complexity of the regular k-Means algorithm for an image of size $N \times N$ pixels is $O(kN^2)$, where k is the number of segments. When we perform the local k-Means algorithm, we do the same as in the k-Means but for image subsquares with added margins. When we fix the margin's size as 0, the computation time for the local k-Means is the same as for the k-Means method with the same number of segments. Therefore, the proposed version is faster because we use only 3 segments instead of 8 ones. On the other hand, when we fix the margin's size as large as the square size, we obtain the mask of size 9 times larger than this square. So, the computation time is $9kN^2$ and the time

complexity is $O(kN^2)$, the same as for the k-Means algorithm. In practical applications, the computation times are comparable between these two methods.

6. Summary

Unlike real images, for which edge detection is relatively well implemented these days, focused images can be troublesome. And it applies not only to really focused images but also to images with different levels of sharp edges. This also can take place in cases when one or a few objects are in motion. In this paper, we presented a new algorithm that can efficiently detect both sharp and smooth edges in focused images alike. The commonly used methods are defined just to use step edges and usually cannot cope with smooth edges properly. The proposed method was defined to overcome the limitations of the existing methods. This improvement was achieved by applying the k-Means algorithm locally. This action causes the algorithm to adapt to the image content so it can correctly detect the edge, no matter if it is sharp or smooth. The proposed method was compared to the reference methods that were chosen as quite different: the classical pointwise one, the modern geometrical one, and the intelligent ML-based one. In all cases, this method gave better results of edge detection than the existing ones.

It is worth mentioning that, although there are plenty of edge detection methods in use, it is not so important which methods we compare our results, since all of them are defined on the assumption that an edge is a step discontinuity. In this paper, we assumed that an edge can also be smooth and has to be detected as well. Thus, this is the main strength of the proposed method. However, this approach raises a number of open problems with the definition of edges with varying smoothness and the proper evaluation of the detection of such edges. Therefore, our future work is to build a model of an edge with varying smoothness and find a way to evaluate such edge detection objectively.

Funding: This research received no external funding.

Conflicts of Interest: The author declare no conflict of interest.

References

1. Tovee, M.J. *An Introduction to the Visual System*; Cambridge University Press: Cambridge, UK, 2008.
2. Sun, R.; Lei, T.; Chen, Q.; Wang, Z.; Du, X.; Zhao, W.; Nandi, A.K. Survey of Image Edge Detection. *Front. Signal Process.* **2022**, *2*. [CrossRef]
3. Ziou, D.; Tabbone, S. Edge Detection Techniques-An Overview. *Pattern Recognit. Image Ann.* **1998**, *8*, 537–559.
4. Prewitt, J.M. Object enhancement and extraction. *Pict. Process. Psychopictor.* **1970**, *10*, 15–19.
5. Shrivakshan, G.T.; Chandrasekar, C. A Comparison of Various Edge Detection Techniques Used in Image Processing. *Int. J. Comput. Sci. Issues* **2012**, *9*, 269.
6. Sobel, I.; Feldman, G. A 3 × 3 isotropic gradient operator for image processing. In *A Talk at the Stanford Artificial Project*; Stanford University: Stanford, CA, USA, 1968; pp. 271–272.
7. Sobel, I. Camera Models and Machine Perception. Ph.D. Thesis, Electrical Engineering Department, Stanford University, Stanford, CA, USA, 1970.
8. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *8*, 679–698. [CrossRef] [PubMed]
9. Marr, D.; Hildreth, E. Theory of Edge Detection. *Proc. R. Soc. Lond.* **1980**, *207*, 187–217. [PubMed]
10. Wang, X. Laplacian Operator-Based Edge Detectors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 886–890. [CrossRef]
11. Kang, H.; Lee, S.; Chui, C.K. Coherent Line Drawing. In Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering, San Diego, CA, USA, 4–5 August 2007; pp. 43–50.
12. Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *33*, 898–916. [CrossRef]
13. Al Darwich, R.; Babout, L.; Strzecha, K. An Edge Detection Method Based on Local Gradient Estimatio. Application to High-Temperature Metallic Droplet Images. *Appl. Sci.* **2022**, *12*, 6976. [CrossRef]
14. Hough, P.V.C. Method and Means for Recognizing Complex Patterns. U.S. Patent 3069654, 18 December 1962.
15. Popovici, I.; Withers, W.D. Custom-build moments for edge location. *IEEE Tans. Pattern Anal. Mach. Intell.* **2006**, *28*, 637–642. [CrossRef]
16. Lisowska, A. Multiscale moments-based edge detection I. In Proceedings of the EUROCON '09 Conference, St. Petersburg, Russia, 18–23 May 2009; pp. 1414–1419.

17. Yi, S.; Labate, D.; Easley, G.R.; Krim, H. A Shearlet Approach to Edge Analysis and Detection. *IEEE Trans. Image Process.* **2009**, *8*, 929–941.
18. Lisowska, A. Edge Detection by Sliding Wedgelets. In *Lecture Notes in Computer Science*; Part 1; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6753, pp. 50–57.
19. Lisowska, A. Geometrical *Multiresolution Adaptive Transforms. Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2014.
20. Criminisi, A.; Shotton, J.; Konukoglu, E. Decision Forest. A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-supervised Learning. *Found. Trends Comput. Graph. Vis.* **2012**, *7*, 81–227. [CrossRef]
21. Dollar, P.; Zitnick, C.L. Structured Forests for Fast Edge Detection. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; Volume 1, pp. 1841–1848.
22. Hallman, S.; Fowlke, C.C. Oriented edge forests for boundary detection. In Proceedings of the the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Boston, MA, USA, 7–12 June 2015; pp. 1732–1740.
23. Walaa, K.; Kadhum, A.M.; Noor, H. A Novel Edge Detection Method Using K-Means Clustering. *J. Eng. Sustain. Dev.* **2016**, *20*, 207–215.
24. Kapil, S.; Chawla, M.; Ansari, M.D. On K-means data clustering algorithm with genetic algorithm. In Proceedings of the 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 22–24 December 2016; pp. 202–206.
25. Zheng, X.; Lei, Q.; Yao, R.; Gong, Y.; Yin, Q. Image Segmentation Based on Adaptive k-Means Algorithm. *EURASIP J. Image Video Process.* **2018**, *2018*, 68. [CrossRef]
26. Lim, J.J.; Zitnick, C.L.; Dollar, P.; Sketch, T. A Learned Mid-level Representation for Contour and Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3158–3165.
27. Mathur, N.; Dadheech, P.; Gupta, M.K. The K-means Clustering Based Fuzzy Edge Detection Technique on MRI Images. In Proceedings of the International Conference on Advances in Computing and Communications (ICACC), Kochi, India, 2–4 September 2015.
28. Bertasius, G.; Shi, J.; Torresani, L. Deepedg: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4380–4389.
29. Xie, S.; Tu, Z. Holistically-Nested Edge Detection. In Proceedings of the ICCV, Santiago, Chile, 7–13 December 2015; pp. 1395–1403.
30. Ganin, Y.; Lempitsky, V. N4-Field. Neural Network Nearest Neighbor Fields for Image Transforms. In Proceedings of the Asian Conference on Computer Vision, Singapore, 1–5 November 2014.
31. Hwang, J.J.; Liu, T.L. Pixel-Wise Deep Learning for Contour Detection. In Proceedings of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, 7–9 May 2015.
32. Yang, J.; Price, B.L.; Cohen, S.; Lee, H.; Yang, M.H. Object contour detection with a fully convolutional encoder-decoder network. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, 27–30 June 2016; pp. 193–202.
33. He, J.; Zhang, S.; Yang, M.; Shan, Y.; Huang, T. Bi-directional cascade network for perceptual edge detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA, 15–20 June 2019; pp. 3828–3837.
34. Andrade-Loarca, H.; Kutyniok, G.; Oktem, O. Shearlets as feature extractor for semantic edge detectio. The model-based and data-driven realm. *Proc. R. Soc.* **2020**, *476*, 20190841. [CrossRef] [PubMed]
35. Su, Z.; Liu, W.; Yu, Z. Pixel Difference Networks for Efficient Edge Detection. In Proceedings of the Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
36. Theodoridis, S.; Koutroumbas, K. *Pattern Recognition*; Academic Press: Cambridge, MA, USA, 2008.
37. Leydier, Y.; Bourgeois, F.; Emptoz, H. Serialized k-Means for Adaptative Color Image Segmentation. Application to Document Images and Others, I. Document Analysis Systems VI DAS 2004. In *Lecture Notes in Computer Science*; Marinai, S., Dengel, A.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3163.
38. The Database of Free Images. Available online: https://www.freeimages.com/ (accessed on 13 January 2022).
39. Schelling, B.; Plant, C. KMN—Removing Noise from K-Means Clustering Results. In *Big Data Analytics and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2018.