Journal of
Sensor and Actuator Networks

MDPI

Article

Region Segmentation of Images Based on a Raster-Scan Paradigm

Luka Lukač, Andrej Nerat, Damjan Strnad, Štefan Horvat and Borut Žalik *

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, SI-2000 Maribor, Slovenia; luka.lukac@um.si (L.L.); andrej.nerat@um.si (A.N.); damjan.strnad@um.si (D.S.); stefan.horvat@um.si (Š.H.)
* Correspondence: borut.zalik@um.si

Abstract: This paper introduces a new method for the region segmentation of images. The approach is based on the raster-scan paradigm and builds the segments incrementally. The pixels are processed in the raster-scan order, while the construction of the segments is based on a distance metric in regard to the already segmented pixels in the neighbourhood. The segmentation procedure operates in linear time according to the total number of pixels. The proposed method, named the RSM (raster-scan segmentation method), was tested on selected images from the popular benchmark datasets MS COCO and DIV2K. The experimental results indicate that our method successfully extracts regions with similar pixel values. Furthermore, a comparison with two of the well-known segmentation methods—Watershed and DBSCAN—demonstrates that the proposed approach is superior in regard to efficiency while yielding visually similar results.

Keywords: image analysis; segment; distance metric; Watershed; DBSCAN

check for updates

Citation: Lukač, L.; Nerat, A.; Strnad, D.; Horvat, Š.; Žalik, B. Region Segmentation of Images Based on a Raster-Scan Paradigm. *J. Sens. Actuator Netw.* **2024**, *13*, 80. https://doi.org/10.3390/jsan13060080

Academic Editor: Lei Shu

Received: 7 October 2024
Revised: 22 November 2024
Accepted: 25 November 2024
Published: 28 November 2024

1. Introduction

Image segmentation denotes a process of dividing an image into a set of non-overlapping regions [1]. It is one of the most basic image preprocessing methods and is utilised in many applications, including object detection [2,3], medical imaging [4–6], remote sensing [7,8], and biometric recognition [9–11]. There are numerous methods available for this task, with the main objective of extracting adjacent pixel groups that share similar features and properties [12]. Different approaches can be used for performing image segmentation, which are divided into three groups [13]:

- Region segmentation.
  Region segmentation methods, also referred to as classic segmentation methods, exploit the similarity between pixels to arrange them into regions [13]. The most common region segmentation methods and their principles are considered briefly in Section 2.
- Collaborative segmentation.
  Collaborative segmentation (or co-segmentation) methods deal with the segmentation of similar objects in multiple correlated images [14]. The concept was introduced in [15], and has been used primarily for tracking the objects in videos [16–18].
- Semantic segmentation.
  The main goal of semantic segmentation methods is to divide an image into meaningful regions that are assigned to the most suitable predefined category labels [19,20]. Nowadays, the great majority of semantic segmentation methods are based on deep neural networks and other machine learning techniques. The most popular architectures include SegNet [21], ReSeg [22], DeepLab [23], CFNet [24], and HRViT [25]. Despite high segmentation performance [26], the downside of such methods is the demand for huge pre-annotated datasets during the training phase of the models.

The existing segmentation methods have proven their strengths in various tasks of image analysis [13]. However, depending on the technique, the available methods have

several disadvantages: inefficiency, troublesome implementation, difficult manual tuning of parameters, and a prolonged training phase (machine learning approaches) [12]. The proposed approach overcomes the issues of other segmentation methods. It is efficient and simple to implement while still yielding comparable results to related region segmentation methods. It is especially suitable for deployment on edge devices [27] (e.g., surveillance cameras) due to its low computational resource demands.
The main contributions of this paper are as follows:

- A new, highly efficient method named the raster-scan segmentation method (RMS) based on a raster scan for the region segmentation of images.
- The design of different distance metrics, which define the similarity of pixels to segments, and actions, which are utilised for the incremental building of segments.
- Extensive experimental work, which demonstrates the efficiency of the RMS, and a comparison with state-of-the-art region segmentation methods (i.e., Watershed and DBSCAN).

The remainder of the paper is structured in the following way. Section 2 highlights the usage of image segmentation in sensor and actuator networks, and summarises the related methods for performing image segmentations. Section 3 describes the proposed method for the segmentation of images using the considered raster-scan approach, the RSM (raster-scan segmentation method). In Section 4, the results of the proposed method are shown and compared with other popular region segmentation methods. Section 5 concludes the paper, summarises the principles of the RSM, and describes future work.

2. Background and Related Work

Today, edge computing is one of the most popular paradigms in Computer Science [28]. The main characteristic of edge computing is that the majority of computations are performed on edge devices with low computational resources [29]. Sensor and actuator networks frequently include edge devices that gather image data. The segmentation methods should be efficient in terms of time and memory consumption in order to enable fast enough performance of various operations on images (tracking the objects in the scene). Image segmentation methods with low computational complexity have been used successfully on different edge devices in sensor and actuator networks, including remote sensing data [8,30,31], handwritten text [20], medical scans [32,33], and footage from surveillance cameras [34,35]. In robotics, image segmentation is often used for the recognition and analysis of objects within the robot's surroundings [36–38].

The proposed raster-scan segmentation method belongs to the region segmentation methods, as its main goal is to split an image into regions with similar pixel values. In Table 1, the most popular related region segmentation methods and their main advantages and disadvantages are summarised. In the continuation of this section, a brief description of each listed segmentation method is provided.

Table 1. Related region segmentation methods.

Method	Advantages	Disadvantages
Differential operators [39]	Low computational complexity	Poor resistance to noise, bad results for images with small pixel gradients
Hough transform [40]	High-accuracy segmentation of regular polygons	Poor segmentation of concave and wavy structures, high computational complexity
Active contour approach [41,42]	Accurate segmentation of concave and wavy polygons	Difficult tuning of parameters
Thresholding [43,44]	Low computational complexity, robust adaptive approaches	Difficult choice of threshold value, possibility of over-segmentation
Watershed [45]	Low computational complexity, efficient marker-controlled approach	Tendency toward over-segmentation, difficult determination of markers' positions
k-means [46–48]	Simplicity, low computational complexity	Difficult tuning of parameters
DBSCAN [49–51]	Accurate segmentation of non-linearly separable image clusters	Sensitivity to input parameters
Superpixel methods [52–54]	Low computational complexity	Inability to process regions with high-intensity varieties

2.1. Edge Detection

Edge detectors are the earliest and most basic methods for region segmentation. Their main goal is to detect pixels with high gradients that represent boundaries of regions [13]. There are several approaches to this task, among which the least complex are differential operators. Their main advantage is low computational complexity. However, their main disadvantage is that they are very prone to producing visually peculiar results if there is even a small amount of noise present in an image [55]. A lot of differential operators have been proposed throughout the years, among which the most popular are Roberts, Kirsch, Deriche, Prewitt, Laplacian, Sobel, and Canny operators [39]. Their time complexity is $O(n \log n)$, where n signifies the total number of pixels in an image.

Hough transform is a popular method for the detection of contours in an image. It was proposed by Duda and Hart [56], and has been adopted for many computer vision tasks, including image segmentation [40,57]. The main advantage of Hough transform is its high accuracy for the detection of regular polygons. On the other hand, it may not produce good segmentation results for concave and wavy structures in an image. Furthermore, Hough transform can be significantly demanding in terms of time complexity when dealing with such structures. In its basic form for detecting lines, the time complexity of Hough transformation is $O(n^4)$ [58].

An iterative active contour approach (also referred to as snakes) was proposed by Kass et al. [59]. It is used for extracting regions from an image by fitting simple polylines to region edges. The latter is performed by energy minimisation [59]: the image forces pull the contour to important image features (i.e., edges, vertices, and lines) while obeying the constraints (either user-given or determined computationally). This method has been proven to yield good results in some cases [41,42]. However, there are two major disadvantages of the active contours method: sensitivity to the initial positions of control points, and difficult initialisation of their input parameters [60]. The time complexity of the basic approach is $O(icm^2)$, where i signifies the number of iterations, c the selected number of control points, and m the number of scales (the contour is fitted to image regions in several resolutions).

Other image segmentation methods based on edge detection have been proposed in the past, such as graph cut optimisations [61,62]. Although they can produce high-accuracy segmentation results, they are only able to divide an image into two regions (foreground and background).

2.2. Region Division

Region division segmentation methods are based either on the serial or the parallel region division [13]. Serial region division splits an image into regions sequentially, while regions are formed simultaneously when parallel region division is applied.

A typical parallel region division technique is thresholding. In the first step, a threshold value is determined either manually or heuristically. After that, each pixel is assigned a binary value [63], based on whether the pixel's value is above or below the threshold. Consequently, the obtained binary image consists of two segments. The described approach has low computational complexity. Division into two regions, however, is not particularly useful for images with many details. Therefore, thresholding with multiple thresholds [64] and adaptive local thresholding [43,44] were proposed in order to increase the usability of this segmentation method in practice. The time complexity of the described method is $O(n)$.

Watershed techniques utilise the geology concept of drainage divides for data segmentation [65,66]. A similar approach, based upon serial region division, is used for region segmentation [67,68]. A greyscale image is interpreted as a 3D topographic relief, which is split into valleys, ridges, and slopes [69]. The main advantage of the Watershed is low time complexity, as it operates in $O(n)$. However, it tends to over-segmentise an image. In order to solve this issue, a marker-controlled Watershed was proposed [45] where significant parts of the image are marked either by a user or using a heuristic.

2.3. Clustering

Clustering is a popular technique for organising data by grouping near data samples [70]. It is also used in different fields of image analysis [71–73]. One of the earliest and most well-known clustering algorithms is k-means [74]. Its main advantage is low computational complexity. On the other hand, major challenges arise concerning the choice of the parameter k that would yield the best segmentation, and the initialisation of the centroid pixels' locations. Therefore, several analyses of an image are usually performed before the segmentation procedure [46–48,75–77]. The time complexity of k-means is $O(knt)$, where t specifies the number of iterations, while k signifies the number of clusters.

Density-based spatial clustering of applications with noise (DBSCAN) was proposed by Ester et al. [78]. Compared to k-means, its major advantage is its ability to process non-linearly separable image structures. DBSCAN has been used successfully for image segmentations [49–51]. Its main disadvantage is having to precisely tune the input parameters in order to obtain the expected results. In the worst case, DBSCAN has a time complexity of $O(n^2)$, although, with the proper data structures, its time complexity is reduced to $O(n \log n)$ [79].

Superpixel segmentation is a clustering method that splits a weighted graph into subgraphs with low coupling and high cohesion [13]. Such an approach has been used in many image segmentation applications [52–54]. Despite the low time complexity using the proper implementation [54], there are several drawbacks of different superpixel segmentation methods: no explicit control over the superpixels' numbers and compactness [52], and failing to correctly segmentise image regions that have high-intensity variety [80]. The time complexity of the most efficient superpixel segmentation methods is $O(n)$ [54].

3. Raster-Scan Segmentation Method

The proposed method is based on a raster-scan paradigm that derives from the era of analogue television [81] based on a cathode-ray tube (CRT). The CRT directs a beam of electrons to the phosphorescent coating on the screen. As a result, a spot of light is produced at a place where the electron beam hits the screen coating. In order to display the picture on the screen, the CRT directs the beam from the left to the right, and, after the whole line is drawn, the procedure is repeated in the next row. This drawing order is known as the scan-line order [82], while the covering of the entire screen in this way is referred to as a raster scan [81].

Let I be a raster image with X columns and Y rows. Consequently, the number of pixels in I is $n = X \cdot Y$. With no loss of generality, I is considered to be greyscale with b bit-planes, yielding 2^b shades of grey. The goal of region image segmentation is to divide I into a set of segments $\mathcal{S} = \{S_i\}, 1 \leq i \leq |\mathcal{S}|$, so that $|\mathcal{S}| \leq n$. However, it is expected that $|\mathcal{S}| \ll n$.

The segmentation of I is performed in the scan-line order, which is why the method is referred to as the raster-scan segmentation method (RSM) in the continuation. The block diagram that represents the sequence of steps in the RSM is displayed in Figure 1.

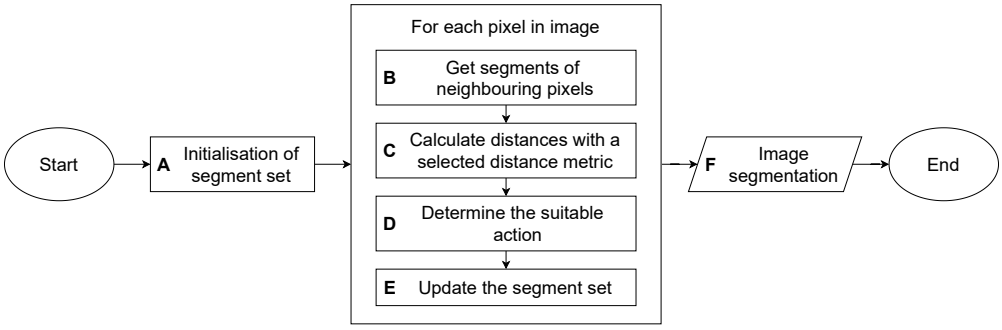


Figure 1. Block diagram of RSM.

In the beginning, the segment set $\mathcal{S}$ is initialised as an empty set (block A in Figure 1). Then, the RSM iterates through I in a scan-line order (see Figure 2). After a pixel $p_{x,y}, 1 \leq x \leq X, 1 \leq y \leq Y$ is reached, segments of neighbouring pixels are obtained (block B). The distances δ are calculated with the selected distance metric d (block C), based on which an action A is determined (block D). According to the local neighbourhood, $p_{x,y}$ is assigned to the segment S_i (already existing or newly created). The partial result of the segmentation is stored in a set of edge-connected segments $\mathcal{S}$ (block E). After all pixels in I are processed, $\mathcal{S}$ contains the final result of the segmentation (block F). The pseudocode at the end of this section gives the details of the algorithm.

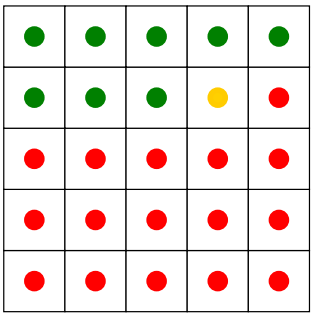


Figure 2. The processing order of the RSM in I. Already processed pixels are marked with green circles, the current pixel is marked with a yellow circle, and non-processed pixels are marked with red circles.

Edge-connected segments are created and built incrementally during the segmentation procedure with the RSM. Each pixel is assigned to a segment S_i. The selection of S_i depends on the already processed pixels, and is based on the selected distance metric d. The following three distance metrics d were used for the calculation of the distance δ:

1. One-Neighbour Metric (Γ_{ON}).
   This metric is based on the differences between the current pixel value $p_{x,y}$ and the pixel values of its left and top neighbours. The differences δ_{left} and δ_{up} are calculated according to Equation (1).

$$\begin{aligned} \delta_{left} &= \left| p_{x,y} - p_{x-1,y} \right| \\ \delta_{up} &= \left| p_{x,y} - p_{x,y-1} \right| \end{aligned} \tag{1}$$

2. Two-Neighbour Metric (Γ_{TN}).
   This metric considers the differences between the current pixel value $p_{x,y}$ and the average value of two pixels to the left and at the top. The differences δ_{left} and δ_{up} are calculated according to Equation (2).

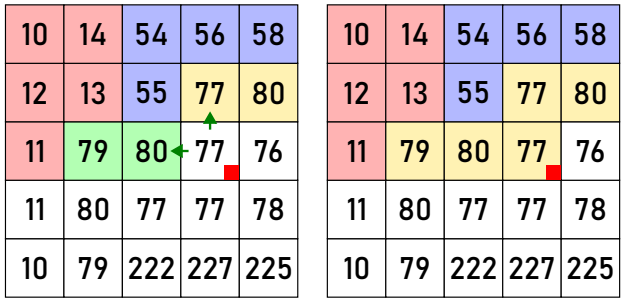
$$\begin{aligned} \delta_{left} &= \left| p_{x,y} - \frac{p_{x-1,y} + p_{x-2,y}}{2} \right| \\ \delta_{up} &= \left| p_{x,y} - \frac{p_{x,y-1} + p_{x,y-2}}{2} \right| \end{aligned} \tag{2}$$

3. Adjacent Segments' Average Metric (Γ_{ASA}).
   Unlike the other two, this metric assesses larger areas in I instead of only separate adjacent pixels' values. Let $p_{x-1,y}$ belong to the segment S_{left} and let $p_{x,y-1}$ belong to S_{up}. If $\overline{S_{left}}$ and $\overline{S_{up}}$ represent the average pixel values of the segments S_{left} and S_{up}, the differences δ_{left} and δ_{up} are calculated according to Equation (3).

$$\begin{aligned} \delta_{left} &= \left| p_{x,y} - \overline{S_{left}} \right| \\ \delta_{up} &= \left| p_{x,y} - \overline{S_{up}} \right| \end{aligned} \tag{3}$$

In order to control the segmentation process, the user needs to provide two parameters. The first parameter is the distance threshold value ϵ, below which $p_{x,y}$ can be added to S_i. The second parameter is the maximum deviation from a segment's average value, Δ_{max}. There are five possible actions A when assigning $p_{x,y}$ to S_i:

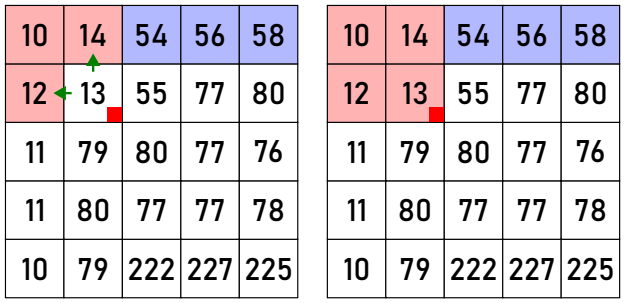
1. *Merge*.
   If $\delta_{left} \leq \epsilon$, $\delta_{left} \leq \Delta_{max}$, $\delta_{up} \leq \epsilon$, $\delta_{up} \leq \Delta_{max}$, and $i_{left} \neq i_{up}$, segments of $p_{x-1,y}$ and $p_{x,y-1}$ are merged into a single segment S_i. After that, S_i is expanded with the current pixel $p_{x,y}$. The latter is performed using a union: $S_i \cup \{p_{x,y}\}$ (Figure 3).



(**a**) (**b**)

Figure 3. The action *Merge*: before (**a**) and after (**b**) the processing of pixel $p_{x,y}$, marked by a red square. The green arrows indicate that $\delta \leq \epsilon$.

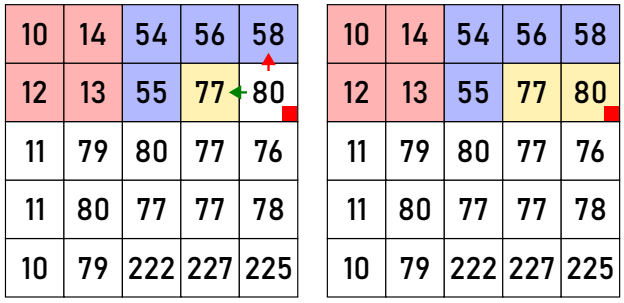
2. *Add*.
   If $\delta_{left} \leq \epsilon$, $\delta_{left} \leq \Delta_{max}$, $\delta_{up} \leq \epsilon$, $\delta_{up} \leq \Delta_{max}$, $i_{left} = i_{up}$, and $p_{x,y-1} \in S_{up}$, the segment S_{up} is expanded as follows: $S_{up} \cup \{p_{x,y}\}$ (Figure 4).



(**a**) (**b**)

Figure 4. The action *Add*: before (**a**) and after (**b**) the processing of pixel $p_{x,y}$, marked by a red square. The green arrows indicate that $\delta \leq \epsilon$.

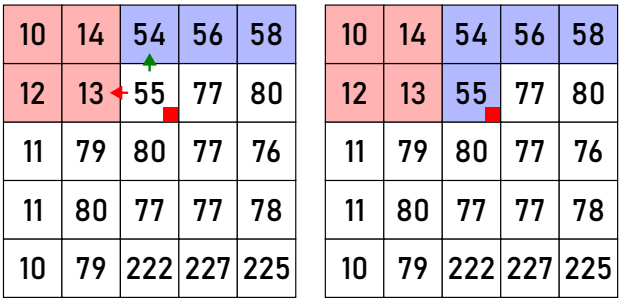
3. *Add-Left*.
   If $\delta_{left} \leq \epsilon$, $\delta_{left} \leq \Delta_{max}$, $\delta_{up} > \epsilon$, and $p_{x-1,y} \in S_{left}$, the segment S_{left} is expanded as follows: $S_{left} \cup \{p_{x,y}\}$ (Figure 5).



(**a**) (**b**)

Figure 5. The action *Add-Left*: before (**a**) and after (**b**) the processing of pixel $p_{x,y}$, marked by a red square. The green arrow indicates that $\delta \leq \epsilon$ while the red arrow indicates that $\delta > \epsilon$.

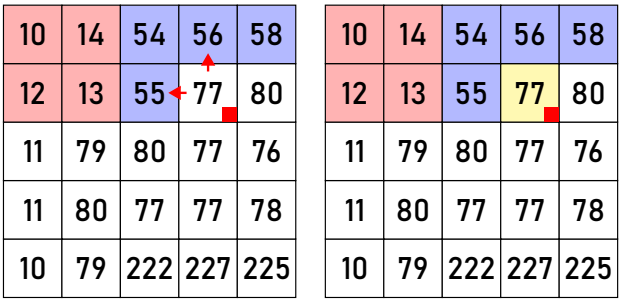
4. *Add-Up*.
   If $\delta_{left} > \epsilon$, $\delta_{up} \leq \epsilon$, $\delta_{up} \leq \Delta_{max}$, and $p_{x,y-1} \in S_{up}$, the segment S_{up} is expanded as follows: $S_{up} \cup \{p_{x,y}\}$ (Figure 6).



(**a**) (**b**)

Figure 6. The action *Add-Up*: before (**a**) and after (**b**) the processing of pixel $p_{x,y}$, marked by a red square. The green arrow indicates that $\delta \leq \epsilon$ while the red arrow indicates that $\delta > \epsilon$.

5. *New*.
   In all other cases, a new segment $S_i = \{p_{x,y}\}$ is added to $\mathcal{S}$ (Figure 7).



(**a**) (**b**)

Figure 7. The action *New*: before (**a**) and after (**b**) the processing of pixel $p_{x,y}$, marked by a red square. The red arrows indicate that $\delta > \epsilon$.

The pseudocode of the RSM is presented in Algorithm 1. In the initialisation phase, an empty set of segments $\mathcal{S}$ is created. In order to enable direct access to the already assigned pixels' segments, a matrix M with indices of segments is used, and is initialised to -1 in the beginning. The loops in Lines 11 and 12 iterate through I. After a pixel $p_{x,y}$ is reached, the index of the left segment i_{left} and the index of the upper segment i_{up} are obtained from M in Lines 13 and 14. Segments S_{left} and S_{up} are obtained from M in Lines 15 and 16. δ_{left} and δ_{up} are calculated in Lines 17 and 18 using the provided d (according to Equations (1), (2), or (3)). In Line 20, one of the five actions A is selected, based on δ_{left}, δ_{up}, ϵ, Δ_{max}, i_{left}, and i_{up}. If the action *Merge* is chosen, the segments of the left and the upper pixels are merged into a new segment, to which the current pixel $p_{x,y}$ is added in Line 24. The newly created S_i is added to $\mathcal{S}$ in Line 25, while the matrix M is updated in Line 26. In the case of the action *Add*, $p_{x,y}$ is added to the common S_i of the left and top pixels in Line 28. After that, $\mathcal{S}$ and M are updated. The actions *Add-Left* and *Add-Top* add $p_{x,y}$ to the left or the top S_i in Line 32 and Line 36, respectively. In both cases, $\mathcal{S}$ and M are then updated accordingly. In the case of the action *New*, a new S_i is created, which is added to $\mathcal{S}$ in Line 41. After that, in Line 42, M is also updated. The final result of the algorithm is a set of segments $\mathcal{S}$.

Algorithm 1 Region image segmentation with the RSM

function RASTER-SCAN-SEGMENTATION-METHOD(I, X, Y, d, ϵ, Δ_{max})
▷ I: an image with the resolution $X \times Y$
▷ d, ϵ: distance metric, distance threshold
▷ Δ_{max}: maximum deviation from the segment's average value
▷ The function returns a set of segments

$\mathcal{S} \leftarrow \{\}$ ▷ Initial set of segments is empty
$M \leftarrow$ InitialiseMatrix$(X, Y, -1)$ ▷ Matrix with indices of segments for pixels
$segmentCounter \leftarrow 0$ ▷ Counter of segments is set to 0 in the beginning

for $y \leftarrow 1$ to Y **do**
for $x \leftarrow 1$ to X **do**
$i_{left} \leftarrow M_{x-1,y}$ ▷ Segment index of the left pixel
$i_{up} \leftarrow M_{x,y-1}$ ▷ Segment index of the upper pixel
$S_{left} \leftarrow$ FindSegment$(M, x-1, y)$
$S_{up} \leftarrow$ FindSegment$(M, x, y-1)$
$\delta_{left} \leftarrow$ CalculateDistance$(d, p_{x,y}, p_{x-1,y}, p_{x-2,y}, S_{left})$ ▷ Equation (1), (2), or (3)
$\delta_{up} \leftarrow$ CalculateDistance$(d, p_{x,y}, p_{x,y-1}, p_{x,y-2}, S_{up})$ ▷ Equation (1), (2), or (3)

$A \leftarrow$ DetermineAction$(\delta_{left}, \delta_{up}, \epsilon, \Delta_{max}, i_{left}, i_{up})$
switch A **do**
case *Merge*:
$segmentCounter \leftarrow segmentCounter + 1$
$S_{segmentCounter} \leftarrow$ MergeSegments$(i_{left}, i_{up}) \cup \{p_{x,y}\}$
AddSegment$(\mathcal{S}, S_{segmentCounter})$
$M_{x,y} \leftarrow segmentCounter$
case *Add*:
$S_{up} \leftarrow S_{up} \cup \{p_{x,y}\}$
UpdateSegment$(\mathcal{S}, S_{up}, i_{up})$
$M_{x,y} \leftarrow i_{up}$
case *Add-Left*:
$S_{left} \leftarrow S_{left} \cup \{p_{x,y}\}$
UpdateSegment$(\mathcal{S}, S_{left}, i_{left})$
$M_{x,y} \leftarrow i_{left}$
case *Add-Up*:
$S_{up} \leftarrow S_{up} \cup \{p_{x,y}\}$
UpdateSegment$(\mathcal{S}, S_{up}, i_{up})$
$M_{x,y} \leftarrow i_{up}$
case *New*:
$segmentCounter \leftarrow segmentCounter + 1$
AddSegment$(\mathcal{S}, \{p_{x,y}\})$
$M_{x,y} \leftarrow segmentCounter$
end switch
end for
end for
return $\mathcal{S}$
end function

4. Results

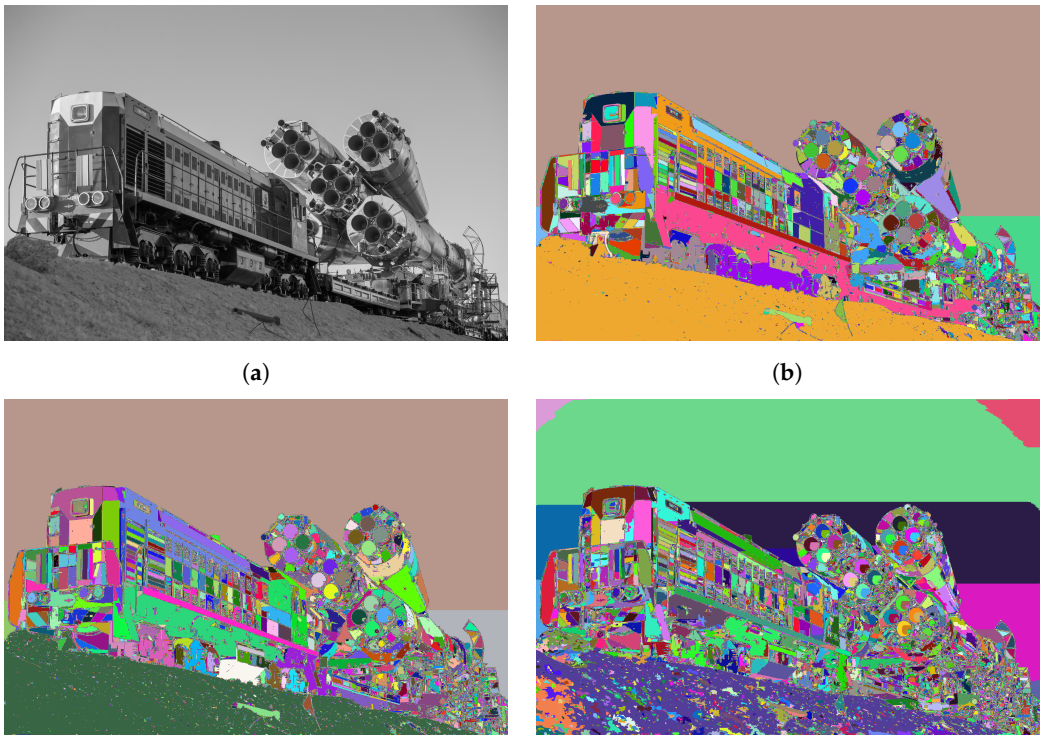
The results of the RSM are presented and discussed in this section. For testing purposes, the popular image datasets MS COCO [83] and DIV2K [84] were used. MS COCO is a well-known image dataset that is often used for the benchmarking of segmentation algorithms [85,86]. It consists of more than 330,000 images. Meanwhile, DIV2K is a consid-

erably smaller dataset as it consists of 1000 images. Primarily, DIV2K is used as a benchmark dataset for super-resolution tasks [87]. In our case, we used both datasets to diversify the test image samples. The performance of the RSM is demonstrated on 10 greyscale images that were selected randomly from MS COCO and DIV2K. The high-resolution images belong to DIV2K, while the images with lower resolutions belong to MS COCO. The names of the selected images, the datasets that they belong to, their resolutions, and the total numbers of pixels n are shown in Table 2.

Table 2. Names, source datasets, resolutions, and total number of pixels of the selected test images I.

I	Dataset	Resolution	n
Airplane	MS COCO	640 × 406	259,840
Baseball	MS COCO	640 × 427	273,280
Canal	DIV2K	2040 × 1524	3,108,960
Fruits	DIV2K	2040 × 1356	2,766,240
Seashore	DIV2K	2040 × 1356	2,766,240
Stop Sign	MS COCO	640 × 480	307,200
Sunflowers	DIV2K	2040 × 1356	2,766,240
Taxi	MS COCO	640 × 480	307,200
Tennis	MS COCO	640 × 427	273,280
Train	DIV2K	2040 × 1356	2,766,240

In Figure 8, the results of image segmentation with the RSM are displayed. Different distance metrics d, described in Section 3, were used. For visualisation purposes, the segments are coloured with randomly selected colours, with the limitation that adjacent segments do not have the same colour. Such colouring enables distinguishing between adjacent segments.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Image segmentation with the RSM using different distance metrics d: (**a**) input image, (**b**) Γ_{ON}, (**c**) Γ_{TN}, (**d**) Γ_{ASA}. Each S_i is coloured with a random colour.

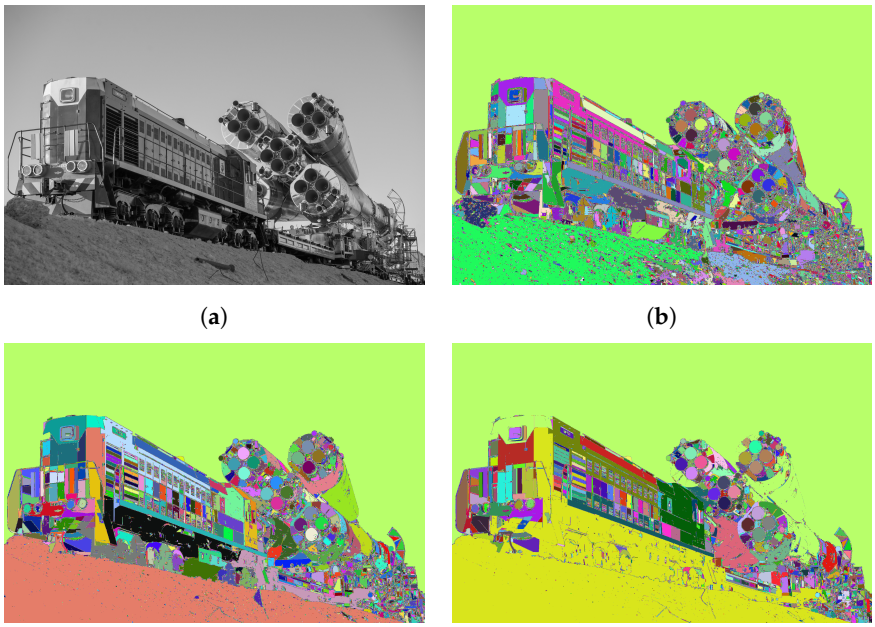
The numbers of segments $|\mathcal{S}|$ for each I using different d values are shown in Table 3. During the segmentation, a suitable action A (*Merge*, *Add*, *Add-Left*, *Add-Up*, or *New*) is selected according to the local neighbourhood of the current pixel $p_{x,y}$. The Γ_{TN} distance

metric produces higher $|\mathcal{S}|$ than Γ_{ON} and Γ_{ASA} for all test images. On the other hand, $|\mathcal{S}|$ is consistently the lowest in all test cases if performing segmentation using Γ_{ON}. In Figure 8b, it can be seen that the Γ_{ON} distance metric produces fewer segments on the ground than Γ_{TN} and Γ_{ASA}. Meanwhile, Γ_{ASA} is able to extract segments that do not contain pixels with large gradients (e.g., segments that represent the sky). Therefore, the most appropriate distance metric d depends on the properties of I. If I contains many edges that are not clearly defined, it is sensible to use Γ_{ASA}. On the other hand, if the pixels on the edges in I have large gradients, Γ_{ON} or Γ_{TN} can yield better results.

Table 3. Number of segments $|\mathcal{S}|$ for test images according to d.

I	Γ_{ON}	Γ_{TN}	Γ_{ASA}
Airplane	23,761	39,949	33,809
Baseball	10,505	16,520	12,659
Canal	215,024	349,251	282,342
Fruits	416,740	698,121	538,066
Seashore	84,695	160,417	118,450
Stop Sign	60,134	74,841	65,451
Sunflowers	349,382	588,149	457,998
Taxi	36,891	55,769	42,701
Tennis	17,703	28,048	25,093
Train	160,412	249,109	190,581

Figure 9 shows region segmentations of the image Train with different values of ϵ (limited to the range that produces sensible segmentation results). The Γ_{ON} distance metric was used for the segmentation. Table 4 shows the number of segments $|\mathcal{S}|$ for the test images according to different values of ϵ. Smaller values of ϵ cause the segmentation to yield higher $|\mathcal{S}|$. Furthermore, by selecting too small a value of ϵ, the RSM tends to over-segment the image (as seen in Figure 9b). Therefore, it is sensible not to choose ϵ values that are close to 0. On the other hand, an ϵ that is too large can lead to a failure in extracting segments with soft edges (e.g., in Figure 9d, where the ground and the bottom part of the train are merged into a single S_i).



(**a**) (**b**)

(**c**) (**d**)

Figure 9. Image segmentation with the RSM according to different values of ϵ: (**a**) input image, (**b**) $\epsilon = 2$, (**c**) $\epsilon = 4$, (**d**) $\epsilon = 6$. Each S_i is coloured with a random colour.

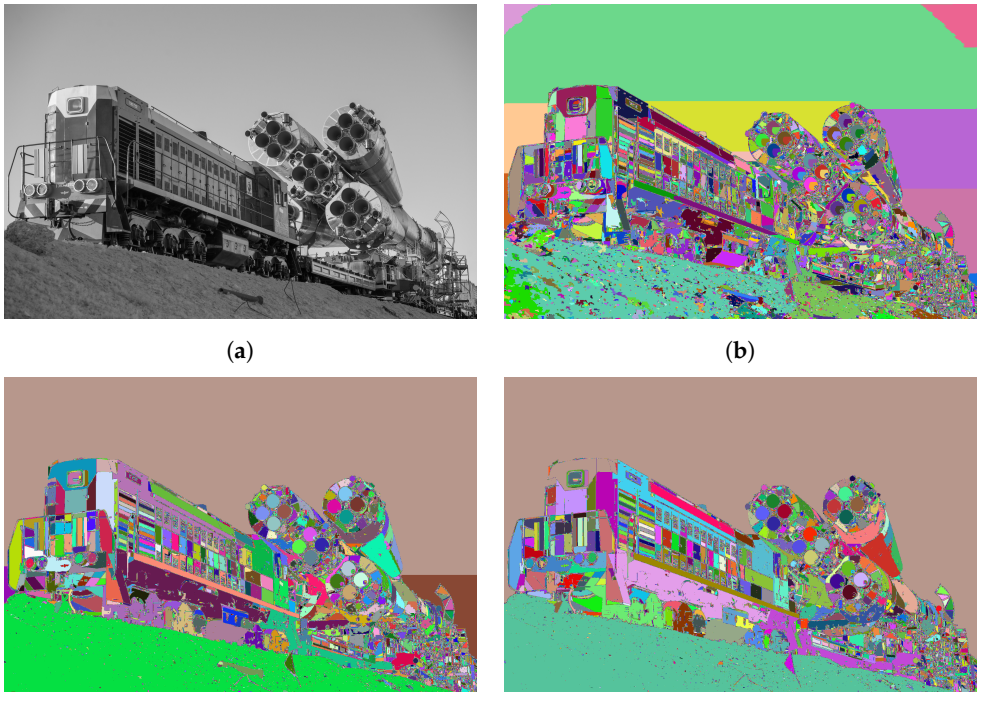
Table 4. Number of segments $|\mathcal{S}|$ for test images according to the parameter ϵ.

I	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	$\epsilon = 5$	$\epsilon = 6$
Airplane	39,469	23,761	16,017	11,789	9245
Baseball	25,948	17,356	13,076	10,505	8669
Canal	443,081	300,451	215,024	161,928	126,380
Fruits	624,147	416,740	294,988	216,854	163,314
Seashore	275,453	180,119	122,328	84,695	59,557
Stop Sign	107,356	87,775	72,513	60,134	50,191
Sunflowers	349,382	224,471	155,674	112,928	84,441
Taxi	86,543	61,972	46,599	36,891	30,081
Tennis	31,942	17,703	12,234	9405	7501
Train	355,059	258,854	200,255	160,412	131,059

Figure 10 displays a comparison between the segmentation results with different values of Δ_{max}. The numbers of segments $|\mathcal{S}|$ for the test images are shown in Table 5. Effectively, small Δ_{max} impacts the segmentation, so that regions with small pixel gradients and large differences in pixel values are split into several segments. Consequently, it can be observed that by increasing Δ_{max}, the number of segments $|\mathcal{S}|$ decreases. The decrease in $|\mathcal{S}|$ is especially noticeable when dealing with smaller values of Δ_{max}.

Table 5. Number of segments $|\mathcal{S}|$ for test images according to parameter Δ_{max}.

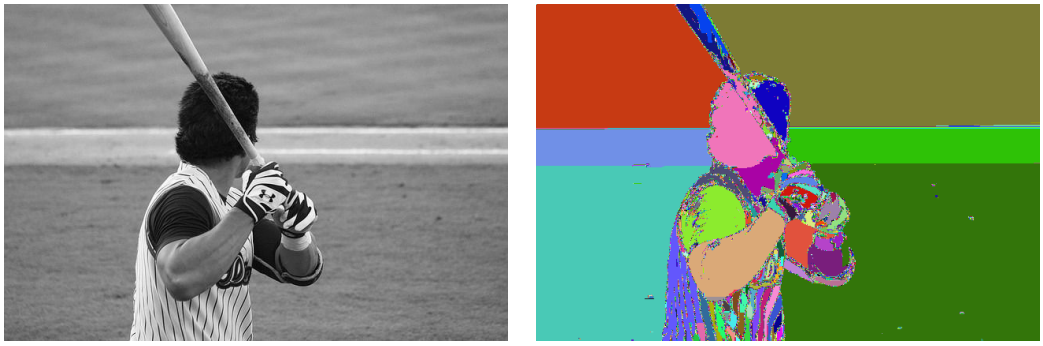
I	$\Delta_{max} = 5$	$\Delta_{max} = 10$	$\Delta_{max} = 20$	$\Delta_{max} = 50$	$\Delta_{max} = 100$
Airplane	27,237	24,581	23,920	23,761	23,747
Baseball	13,126	11,032	10,505	10,360	10,344
Canal	261,231	225,578	216,978	215,024	214,898
Fruits	464,036	429,965	420,911	417,494	416,740
Seashore	124,666	93,531	86,474	84,695	84,444
Stop Sign	67,569	61,315	60,265	60,135	60,134
Sunflowers	366,326	354,651	350,808	349,382	348,971
Taxi	44,238	36,891	35,326	34,930	34,906
Tennis	20,349	18,458	17,754	17,708	17,703
Train	197,737	167,779	160,412	158,535	158,417



(**a**) (**b**) (**c**) (**d**)

Figure 10. Image segmentation with RSM according to different values of Δ_{max}: (**a**) input image, (**b**) $\Delta_{max} = 5$, (**c**) $\Delta_{max} = 20$, (**d**) $\Delta_{max} = 50$. Each S_i is coloured with a random colour.

Examples of region image segmentations of the test images Baseball, Stop Sign, and Taxi are displayed in Figures 11, Figure 12, and Figure 13, respectively. After the segmentation is performed, it can be seen that regions with low variance in pixel values form common segments while the important image details, such as edges and vertices, are preserved. Following this, the parameters d, ϵ, and Δ_{max} were tuned manually, so that the segmentation yielded the best visual results. Although the majority of the regions were detected as expected, we can observe that some parts of the image are over-segmented (especially noticeable in Figure 12 below the Stop Sign).



(**a**) (**b**)

Figure 11. Region segmentation of Baseball: (**a**) input image, (**b**) segmented image.

(**a**) (**b**)

Figure 12. Region segmentation of Stop Sign: (**a**) input image, (**b**) segmented image.

(**a**) (**b**)

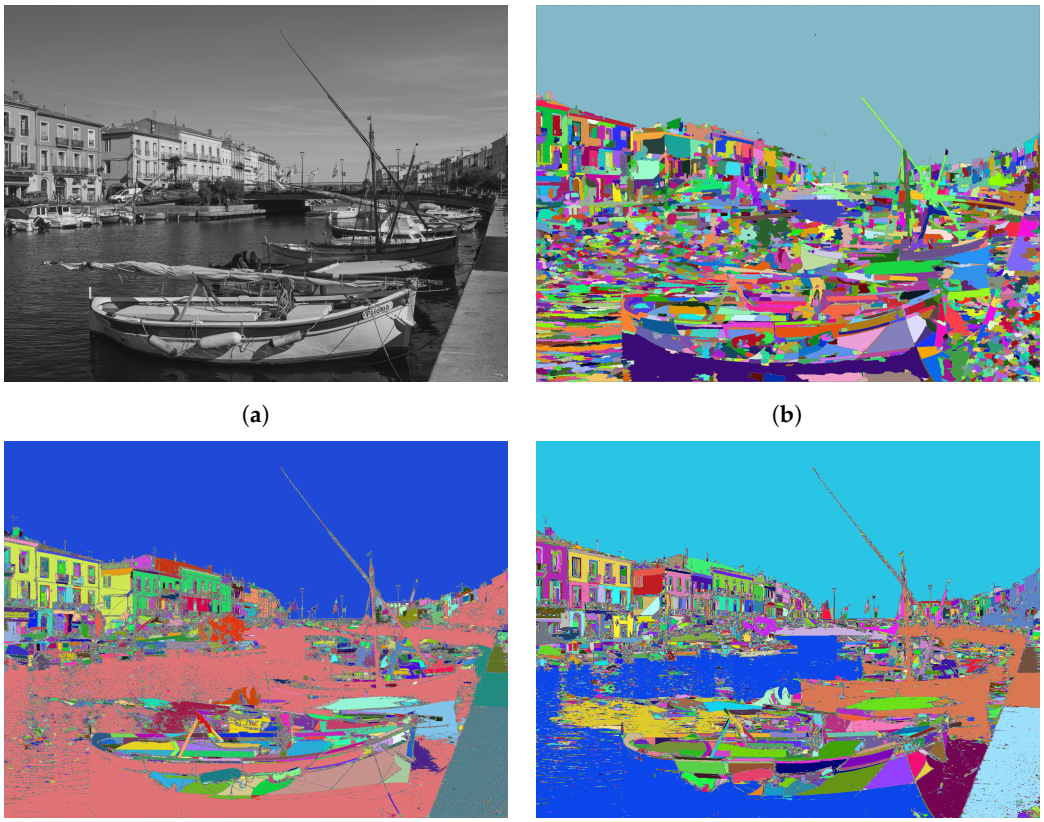
Figure 13. Region segmentation of Taxi: (**a**) input image, (**b**) segmented image.

In Figure 14, different region segmentation methods are compared on the test image Canal. Besides the RSM, we used the marker-controlled Watershed method from the

library OpenCV [88] and DBSCAN from a popular C++ library mlpack [89]. In Table 6, the numbers of segments $|\mathcal{S}|$ are summarised for the test image Canal. Despite the fact that Watershed yielded the fewest segments among the three methods, I was over-segmented in some areas (e.g., the water surface). On the other hand, the result of DBSCAN was considerably better, as larger areas are grouped to form segments. The RSM produces visually similar results to DBSCAN.

Table 6. Number of segments $|\mathcal{S}|$ using different segmentation methods.

I	Watershed	DBSCAN	RSM
Airplane	28,409	48,542	23,761
Baseball	23,278	21,298	10,505
Canal	197,729	498,016	215,024
Fruits	243,714	583,656	416,740
Seashore	126,108	394,168	84,695
Stop Sign	23,324	113,411	60,134
Sunflowers	157,878	576,622	349,382
Taxi	27,862	96,811	36,891
Tennis	24,391	42,800	17,703
Train	153,704	538,013	160,412



(**a**) (**b**)

(**c**) (**d**)

Figure 14. Comparison of different region segmentation methods: (**a**) input image, (**b**) Watershed, (**c**) DBSCAN, (**d**) RSM.

In terms of theoretical time complexity, with n denoting the total number of pixels, the worst case for Watershed segmentation is $O(n \log n)$. On average, though, the algorithm performs in linear time $O(n)$ [90]. DBSCAN's worst time complexity using proper implementation is $O(n \log n)$. The RSM processes each pixel only once, which is performed in linear time $O(n)$. Each classified pixel is assigned to a certain segment whose index is stored in a 2D matrix. Consequently, a query for a segment during the segmentation procedure is performed in $O(1)$. The operation of merging can be implemented with the

union-find data structure, which has the amortised time complexity of $O(1)$ [91]. Based on this, it can be concluded that the time complexity of the RSM is $O(n)$.

The average elapsed CPU times of region segmentation on the test images with Watershed, DBSCAN, and the RSM are shown in Table 7. The experiments were conducted on a personal computer with an Intel Core i9-12900K CPU, 64 GB of DDR5 RAM, and Windows 11 OS. Watershed's performance turned out to be almost on a par with the RSM, while DBSCAN was considerably slower for images with large numbers of pixels. Despite the good performance of Watershed, the RSM was faster in all the test cases.

Table 7. Time measurements of different region segmentation methods on test images. The measurements that are bold indicate the method that performed the fastest for a given test image.

I	Watershed [s]	DBSCAN [s]	RSM [s]
Airplane	0.011	0.743	**0.006**
Baseball	0.011	0.916	**0.005**
Canal	0.098	17.163	**0.069**
Fruits	0.136	10.462	**0.083**
Seashore	0.076	16.246	**0.053**
Stop Sign	0.011	0.818	**0.010**
Sunflowers	0.103	13.694	**0.075**
Taxi	0.012	0.837	**0.010**
Tennis	0.011	0.879	**0.008**
Train	0.080	13.883	**0.053**

5. Conclusions

This paper introduces a new method, the raster-scan segmentation method (RSM), for the region segmentation of images based on a raster scan. During the segmentation in scan-line order, image segments are built incrementally according to the similarity in the pixels' local neighbourhoods. Different distance metrics are used to estimate the similarity between pixels. The RSM was extensively tested on images from the public image datasets MS COCO and DIV2K. Five images were selected randomly from each dataset, and were used to demonstrate the results of the proposed method. The RSM was compared with the widely used region segmentation methods Watershed and DBSCAN. Extensive experimental work revealed that the RSM brings several advantages. Besides its ability to extract segments with pixels that have small gradients, the RSM demonstrated comparable results to other state-of-the art region segmentation methods while being faster in terms of elapsed CPU time.

Region image segmentation could be used as a preprocessing step in different applications of image processing. The RSM could be applied on real-world datasets, or could be used as a real-time segmentation method in sensor and actuator networks due to its low computational complexity. Another possible use case would be remote sensing. Our method could be used as an alternative to other segmentation methods in order to extract different landscape elements from satellite images. Lastly, a possible domain where the RSM could be applied is object-based image compression, where compression could be performed on extracted segments instead of in the pixel space.

There are several ways to implement improvements to the RSM in the future. To start with, the RSM has no mechanisms to deal with noise present in an image. Additional studies on various types and intensities of noise, applied on images, should be conducted. After that, the most suitable noise-removing methods in the preprocessing step should be identified. In addition, currently, the method's parameters are tuned manually, which can be inefficient in real-life applications. Consequently, an automatic approach for selecting the parameters, based on the extracted image's characteristics, should be developed.

Author Contributions: Conceptualization, L.L. and B.Ž.; methodology, L.L., A.N., D.S. and B.Ž.; software, L.L. and Š.H.; validation, L.L., A.N. and B.Ž.; formal analysis, D.S. and B.Ž.; investigation, A.N. and D.S.; resources, Š.H. and B.Ž.; data curation, L.L. and Š.H.; writing—original draft preparation, L.L.; writing—review and editing, L.L., Š.H. and B.Ž.; visualization, L.L.; supervision, B.Ž. and D.S.; project administration, B.Ž.; funding acquisition, B.Ž. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovenian Research and Innovation Agency under Research Project J2-4458 and Research Programme P2-0041.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code of the image segmentation software is available at https://github.com/luckyLukac/ScanLineImageSegmentation, accessed on 30 September 2024. The datasets MS COCO and DIV2K are available at https://cocodataset.org/ and https://data.vision.ee.ethz.ch/cvl/DIV2K, respectively (accessed on 30 September 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cheng, H.; Jiang, X.; Sun, Y.; Wang, J. Color image segmentation: Advances and prospects. *Pattern Recognit.* **2001**, *34*, 2259–2281. [CrossRef]
2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 142–158. [CrossRef]
3. Hoeser, T.; Kuenzer, C. Object detection and image segmentation with deep learning on Earth observation data: A review-Part I: Evolution and recent trends. *Remote Sens.* **2020**, *12*, 1667. [CrossRef]
4. Pham, D.L.; Xu, C.; Prince, J.L. Current methods in medical image segmentation. *Annu. Rev. Biomed. Eng.* **2000**, *2*, 315–337. [CrossRef] [PubMed]
5. Fenster, A.; Chiu, B. Evaluation of Segmentation Algorithms for Medical Imaging. In Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, 1–4 September 2005; pp. 7186–7189. [CrossRef]
6. Guo, Z.; Li, X.; Huang, H.; Guo, N.; Li, Q. Deep learning-based image segmentation on multimodal medical imaging. *IEEE Trans. Radiat. Plasma. Med. Sci.* **2019**, *3*, 162–169. [CrossRef] [PubMed]
7. Wang, Z.; Jensen, J.R.; Im, J. An automatic region-based image segmentation algorithm for remote sensing applications. *Environ. Modell. Softw.* **2010**, *25*, 1149–1165. [CrossRef]
8. Kotaridis, I.; Lazaridou, M. Remote sensing image segmentation advances: A meta-analysis. *ISPRS J. Photogramm. Remote Sens.* **2021**, *173*, 309–322. [CrossRef]
9. El-Hajj-Chehade, W.; Abdel Kader, R.; Kassem, R.; El-Zaart, A. Image Segmentation for Fingerprint Recognition. In Proceedings of the 2018 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, India, 7–9 December 2018; pp. 314–319. [CrossRef]
10. Tobji, R.; Di, W.; Ayoub, N. FMnet: Iris segmentation and recognition by using fully and multi-scale CNN for biometric security. *Appl. Sci.* **2019**, *9*, 2042. [CrossRef]
11. Trokielewicz, M.; Czajka, A.; Maciejewicz, P. Post-mortem iris recognition with deep-learning-based image segmentation. *Image Vision Comput.* **2020**, *94*, 103866. [CrossRef]
12. Kaur, D.; Kaur, Y. Various image segmentation techniques: A review. *Int. J. Comput. Sci. Mob. Comput.* **2014**, *3*, 809–814.
13. Yu, Y.; Wang, C.; Fu, Q.; Kou, R.; Huang, F.; Yang, B.; Yang, T.; Gao, M. Techniques and challenges of image segmentation: A review. *Electronics* **2023**, *12*, 1199. [CrossRef]
14. Vicente, S.; Rother, C.; Kolmogorov, V. Object Cosegmentation. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 2217–2224. [CrossRef]
15. Rother, C.; Minka, T.; Blake, A.; Kolmogorov, V. Cosegmentation of Image Pairs by Histogram Matching—Incorporating a Global Constraint into MRFs. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 17–22 June 2006; pp. 993–1000. [CrossRef]
16. Chen, D.J.; Chen, H.T.; Chang, L.W. Video Object Cosegmentation. In Proceedings of the 20th ACM International Conference on Multimedia, Nara, Japan, 29 October–2 November 2012; pp. 805–808. [CrossRef]
17. Liu, Z.; Wang, L.; Hua, G.; Zhang, Q.; Niu, Z.; Wu, Y.; Zheng, N. Joint video object discovery and segmentation by coupled dynamic Markov networks. *IEEE Trans. Image Process.* **2018**, *27*, 5840–5853. [CrossRef]
18. Wang, L.; Lv, X.; Zhang, Q.; Niu, Z.; Zheng, N.; Hua, G. Object cosegmentation in noisy videos with multilevel hypergraph. *IEEE Trans. Multimedia* **2021**, *23*, 1287–1300. [CrossRef]
19. Hao, S.; Zhou, Y.; Guo, Y. A brief survey on semantic segmentation with deep learning. *Neurocomputing* **2020**, *406*, 302–321. [CrossRef]

20. Patil, S.; Varadarajan, V.; Mahadevkar, S.; Athawade, R.; Maheshwari, L.; Kumbhare, S.; Garg, Y.; Dharrao, D.; Kamat, P.; Kotecha, K. Enhancing optical character recognition on images with mixed text using semantic segmentation. *J. Sens. Actuator Netw.* **2022**, *11*, 63. [CrossRef]
21. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
22. Visin, F.; Ciccone, M.; Romero, A.; Kastner, K.; Cho, K.; Bengio, Y.; Matteucci, M.; Courville, A. ReSeg: A Recurrent Neural Network-Based Model for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, USA, 26 June–1 July 2016.
23. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef] [PubMed]
24. Zhang, H.; Zhang, H.; Wang, C.; Xie, J. Co-Occurrent Features in Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 548–557.
25. Gu, J.; Kwon, H.; Wang, D.; Ye, W.; Li, M.; Chen, Y.H.; Lai, L.; Chandra, V.; Pan, D.Z. Multi-Scale High-Resolution Vision Transformer for Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 12094–12103. [CrossRef]
26. Kulhandjian, H.; Barron, J.; Tamiyasu, M.; Thompson, M.; Kulhandjian, M. AI-based pedestrian detection and avoidance at night using multiple sensors. *J. Sens. Actuator Netw.* **2024**, *13*, 34. [CrossRef]
27. Chen, J.; Ran, X. Deep learning with edge computing: A review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
28. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
29. Nayak, S.; Patgiri, R.; Waikhom, L.; Ahmed, A. A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications. *Digit. Commun. Netw.* **2024**, *10*, 783–804. [CrossRef]
30. Hossain, M.D.; Chen, D. Segmentation for Object-Based Image Analysis (OBIA): A review of algorithms and challenges from remote sensing perspective. *ISPRS J. Photogramm. Remote Sens.* **2019**, *150*, 115–134. [CrossRef]
31. Jahanbakht, M.; Xiang, W.; Waltham, N.J.; Azghadi, M.R. Distributed deep learning and energy-efficient real-time image orocessing at the edge for fish segmentation in underwater videos. *IEEE Access* **2022**, *10*, 117796–117807. [CrossRef]
32. Wang, E.K.; Chen, C.M.; Hassan, M.M.; Almogren, A. A deep learning based medical image segmentation technique in Internet-of-Medical-Things domain. *Future Gener. Comput. Syst.* **2020**, *108*, 135–144. [CrossRef]
33. Ali, O.; Ali, H.; Shah, S.A.A.; Shahzad, A. Implementation of a modified U-Net for medical image segmentation on edge devices. *IEEE Trans. Circuits Syst. II Express Br.* **2022**, *69*, 4593–4597. [CrossRef]
34. Gruosso, M.; Capece, N.; Erra, U. Human segmentation in surveillance video with deep learning. *Multimed. Tools Appl.* **2021**, *80*, 1175–1199. [CrossRef]
35. Muhadi, N.A.; Abdullah, A.F.; Bejo, S.K.; Mahadi, M.R.; Mijic, A. Deep learning semantic segmentation for water level estimation using surveillance camera. *Appl. Sci.* **2021**, *11*, 9691. [CrossRef]
36. Bruce, J.; Balch, T.; Veloso, M. Fast and Inexpensive Color Image Segmentation for Interactive Robots. In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), Takamatsu, Japan, 31 October–5 November 2000; pp. 2061–2066. [CrossRef]
37. Wan, H.; Fan, Z.; Yu, X.; Kang, M.; Wang, P.; Zeng, X. A real-time branch detection and reconstruction mechanism for harvesting robot via convolutional neural network and image segmentation. *Comput. Electron. Agric.* **2022**, *192*, 106609. [CrossRef]
38. Podgorelec, D.; Uran, S.; Nerat, A.; Bratina, B.; Pečnik, S.; Dimec, M.; Žaberl, F.; Žalik, B.; Šafarič, R. LiDAR-based maintenance of a safe distance between a human and a robot arm. *Sensors* **2023**, *23*, 4305. [CrossRef] [PubMed]
39. Nixon, M.S.; Aguado, A.S. *Feature Extraction and Image Processing*, 2nd ed.; Academic Press: Oxford, UK, 2013.
40. Ding, Y.; Ping, X.; Hu, M.; Wang, D. Range image segmentation based on randomized Hough transform. *Pattern Recognit. Lett.* **2005**, *26*, 2033–2041. [CrossRef]
41. Sagiv, C.; Sochen, N.; Zeevi, Y. Integrated active contours for texture segmentation. *IEEE Trans. Image Process.* **2006**, *15*, 1633–1646. [CrossRef] [PubMed]
42. Zhang, K.; Zhang, L.; Song, H.; Zhou, W. Active contours with selective local or global segmentation: A new formulation and level set method. *Image Vision Comput.* **2010**, *28*, 668–676. [CrossRef]
43. Jiang, X.; Mojon, D. Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 131–137. [CrossRef]
44. Phansalkar, N.; More, S.; Sabale, A.; Joshi, M. Adaptive Local Thresholding for Detection of Nuclei in Diversity Stained Cytology Images. In Proceedings of the 2011 International Conference on Communications and Signal Processing, Kerala, India, 10–12 February 2011; pp. 218–220. [CrossRef]
45. Benson, C.C.; Rajamani, K.; Lajish, V.L. A review on automatic marker identification methods in watershed algorithms used for medical image segmentation. *Int. J. Innov. Sci. Eng. Technol.* **2015**, *2*, 829–839.
46. Dhanachandra, N.; Manglem, K.; Chanu, Y.J. Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. *Procedia Comput. Sci.* **2015**, *54*, 764–771. [CrossRef]

47. Zheng, X.; Lei, Q.; Yao, R.; Gong, Y.; Yin, Q. Image segmentation based on adaptive K-means algorithm. *J. Image Video Proc.* **2018**, *2018*, 1–10. [CrossRef]
48. Debelee, T.G.; Schwenker, F.; Rahimeto, S.; Yohannes, D. Evaluation of modified adaptive K-means segmentation algorithm. *Comput. Vis. Media.* **2019**, *5*, 347–361. [CrossRef]
49. Manavalan, R.; Thangavel, K. TRUS Image Segmentation Using Morphological Operators and DBSCAN Clustering. In Proceedings of the 2011 World Congress on Information and Communication Technologies, Mumbai, India, 11–14 December 2011; pp. 898–903. [CrossRef]
50. Baselice, F.; Coppolino, L.; D'Antonio, S.; Ferraioli, G.; Sgaglione, L. A DBSCAN Based Approach for Jointly Segment and Classify Brain MR Images. In Proceedings of the 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, Italy, 25–29 August 2015; pp. 2993–2996. [CrossRef]
51. Kurumalla, S.; Rao, P.S. K-nearest neighbor based DBSCAN clustering algorithm for image segmentation. *J. Theor. Appl. Inf. Technol.* **2016**, *92*, 395–402.
52. Comaniciu, D.; Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 603–619. [CrossRef]
53. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
54. Li, Z.; Chen, J. Superpixel Segmentation Using Linear Spectral Clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1356–1363.
55. Muñoz, X.; Freixenet, J.; Cufı, X.; Martı, J. Strategies for image segmentation combining region and boundary information. *Pattern Recognit. Lett.* **2003**, *24*, 375–392. [CrossRef]
56. Duda, R.O.; Hart, P.E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM* **1972**, *15*, 11–15. [CrossRef]
57. Tian, Q.C.; Pan, Q.; Cheng, Y.M.; Gao, Q.X. Fast Algorithm and Application of Hough Transform in Iris Segmentation. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2004; pp. 3977–3980. [CrossRef]
58. Asano, T.; Katoh, N. Variants for the Hough transform for line detection. *Comput. Geom.* **1996**, *6*, 231–252. [CrossRef]
59. Kass, M.; Witkin, A.; Terzopoulos, D. Snakes: Active contour models. *Int. J. Comput. Vision* **1988**, *1*, 321–331. [CrossRef]
60. Xu, G.; Segawa, E.; Tsuji, S. Robust active contours with insensitive parameters. *Pattern Recognit.* **1994**, *27*, 879–884. [CrossRef]
61. Freedman, D.; Zhang, T. Interactive Graph Cut Based Segmentation with Shape Priors. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; pp. 755–762. [CrossRef]
62. Boykov, Y.; Funka-Lea, G. Graph cuts and efficient ND image segmentation. *Int. J. Comput. Vision* **2006**, *70*, 109–131. [CrossRef]
63. Shapiro, L.G.; Stockman, G.C. *Computer Vision*; Prentice Hall: Hoboken, NJ, USA, 2001.
64. Reddi, S.S.; Rudin, S.F.; Keshavan, H.R. An optimal multiple threshold scheme for image segmentation. *IEEE Trans. Syst. Man Cybern.* **1984**, *SMC-14*, 661–665. [CrossRef]
65. Mongus, D.; Žalik, B. An efficient approach to 3D single tree-crown delineation in LiDAR data. *ISPRS J. Photogramm. Remote Sens.* **2015**, *108*, 219–233. [CrossRef]
66. Forte, A.M.; Whipple, K.X. Criteria and tools for determining drainage divide stability. *Earth Planet. Sci. Lett.* **2018**, *493*, 102–117. [CrossRef]
67. Vincent, L.; Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 583–598. [CrossRef]
68. Meijster, A.; Roerdink, J.B.T.M. A Proposal for the Implementation of a Parallel Watershed Algorithm. In Proceedings of the Computer Analysis of Images and Patterns, Prague, Czech Republic, 6–8 September 1995; pp. 790–795. [CrossRef]
69. Kornilov, A.S.; Safonov, I.V. An overview of watershed algorithm implementations in open source libraries. *J. Imaging* **2018**, *4*, 123. [CrossRef]
70. El Habib Kahla, M.; Beggas, M.; Laouid, A.; Hammoudeh, M. A nature-inspired partial distance-based clustering algorithm. *J. Sens. Actuator Netw.* **2024**, *13*, 36. [CrossRef]
71. Žalik, K.R.; Žalik, B. A sweep-line algorithm for spatial clustering. *Adv. Eng. Softw.* **2009**, *40*, 445–451. [CrossRef]
72. Lukač, N.; Žalik, B.; Žalik, K.R. Sweep-hyperplane clustering algorithm using dynamic model. *Informatica* **2014**, *25*, 563–580. [CrossRef]
73. Mittal, M.; Goyal, L.M.; Hemanth, D.J.; Sethi, J.K. Clustering approaches for high-dimensional databases: A review. *WIREs Data. Mining. Knowl. Discov.* **2019**, *9*, e1300. [CrossRef]
74. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1965 and 27 December 1965–7 January 1966 ; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
75. Luo, M.; Ma, Y.F.; Zhang, H.J. A Spatial Constrained K-means Approach to Image Segmentation. In Proceedings of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, Proceedings of the 2003 Joint, Singapore, 15–18 December 2003; pp. 738–742. [CrossRef]
76. Žalik, K.R. An efficient k'-means clustering algorithm. *Pattern Recognit. Lett.* **2008**, *29*, 1385–1391. [CrossRef]

77. Mat Isa, N.A.; Salamah, S.A.; Ngah, U.K. Adaptive fuzzy moving K-means clustering algorithm for image segmentation. *IEEE Trans. Consum. Electron.* **2009**, *55*, 2145–2153. [CrossRef]
78. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
79. Ali, T.; Asghar, S.; Sajid, N.A. Critical Analysis of DBSCAN Variations. In Proceedings of the 2010 International Conference on Information and Emerging Technologies, Karachi, Pakistan, 14–16 June 2010; pp. 1–6. [CrossRef]
80. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient graph-based image segmentation. *Int. J. Comput. Vis.* **2004**, *59*, 167–181. [CrossRef]
81. Jones, P.F. The graphics screen. In *CAD/CAM: Features, Applications and Management*; Macmillan Education: London, UK, 1992; pp. 21–30. [CrossRef]
82. Martin, A. Cathode ray tubes for industrial and military applications. *Adv. Imaging Electron Phys.* **1986**, *67*, 183–328. [CrossRef]
83. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the Computer Vision–ECCV 2014, Springer, Zurich, Switzerland, 6–12 September 2014; pp. 740–755. [CrossRef]
84. Agustsson, E.; Timofte, R. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 1122–1131. [CrossRef]
85. Zhang, H.; Tian, Y.; Wang, K.; Zhang, W.; Wang, F.Y. Mask SSD: An effective single-stage approach to object instance segmentation. *IEEE Trans. Image Process.* **2020**, *29*, 2078–2093. [CrossRef]
86. Masubuchi, S.; Watanabe, E.; Seo, Y.; Okazaki, S.; Sasagawa, T.; Watanabe, K.; Taniguchi, T.; Machida, T. Deep-learning-based image segmentation integrated with optical microscopy for automatically searching for two-dimensional materials. *NPJ 2D Mater. Appl.* **2020**, *4*, 3. [CrossRef]
87. Timofte, R.; Agustsson, E.; Van Gool, L.; Yang, M.H.; Zhang, L.; Lim, B.; Son, S.; Kim, H.; Nah, S.; Lee, K.M.; et al. NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Honolulu, HI, USA, 21–26 July 2017.
88. OpenCV: Image Segmentation with Watershed Algorithm. Available online: https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html (accessed on 1 October 2024).
89. Curtin, R.R.; Edel, M.; Shrit, O.; Agrawal, S.; Basak, S.; Balamuta, J.J.; Birmingham, R.; Dutt, K.; Eddelbuettel, D.; Garg, R.; et al. mlpack 4: A fast, header-only C++ machine learning library. *J. Open Source Softw.* **2023**, *8*, 5026. [CrossRef]
90. Bieniek, A.; Moga, A. An efficient watershed algorithm based on connected components. *Pattern Recognit.* **2000**, *33*, 907–916. [CrossRef]
91. Cormen, T.; Leiserson, C.; Rivest, R.; Stein, C. *Introduction to Algorithms*, 4th ed.; MIT Press: London, UK, 2022.