Acceleration of Clustering-Based Superpixel Algorithms with Low Memory Costs

Tse-Wei Chen, Noriyasu Hashiguchi, Masahiro Ariizumi, Kinya Osa, Daisuke Nakashima, Yasuo Fukuda, Shiori Wakino, Shinji Shiraga, and Masami Kato
Digital System Technology Development Headquarters
Canon Inc.
3-30-2, Shimomaruko, Ohta-ku, Tokyo 146-8501, Japan
e-mail: twchen@ieee.org

Abstract—As a pre-processing step of image segmentation, superpixel algorithms are used to produce small, uniform and compact regions, which can be used for region-based image coding, region-based image processing, and object recognition. In order to meet the requirements of real-time applications for embedded computing, it is necessary to reduce the computational costs of superpixel algorithms and increase the processing speed. In this paper, a series of acceleration schemes for superpixels algorithm is proposed. The features and contributions of this work are stated as follows. Firstly, the spatial distances and the color distances are calculated individually, so that the redundant distance computations can be saved. Secondly, by searching the nearest cluster centroids with centroid priority, the nearest clusters can be found at an early stage. Thirdly, the early-termination mechanism can be applied to the search process to speed up the algorithm without decreasing the quality of image segmentation. Fourthly, the storage for label images and distance images is not required since the operations of nearest centroids are processed in the inner loop of the algorithm. The experiments show that the proposed method achieves the same level of performance as the related work with only 75% of distance computations and 33% of memory costs.

I. INTRODUCTION

To divide an image into multiple regions with similar colors, textures, or spatial positions, many kinds of image segmentation algorithms [1], including graph-based algorithms [2], clustering-based algorithms [3], and mode-seeking algorithms [4], are proposed. As a pre-processing step of image segmentation, superpixel algorithms [5] [6] are used to produce small, uniform and compact regions, which can be used for augmented reality [7], tracking [8], salient object detection [9], and salient region detection [10]. In order to meet the requirements of real-time applications for embedded computing, it is necessary to reduce the computational costs of superpixel algorithms and increase the processing speed. Clustering-based superpixels are popular because of the low computational costs and the simplicity of the algorithms [11] [12] [13].

Although clustering-based superpixel algorithms can be accelerated by reducing the search range of cluster centroids, the computational costs of superpixels in high-resolution images are still high. Acceleration schemes for the K-Means clustering algorithm are reported in literatures [14], but the extra memory costs result in the overhead of implementation in embedded systems, where the bandwidth is limited. The SLIC algorithm [13] also requires the storage of label images and distance images, which is proportional to the image size. In order to handle high-resolution images in embedded systems, it is necessary to speed up the algorithms with low-memory costs. To alleviate the problem, a series of acceleration schemes for clustering-based superpixels algorithm is proposed. In the clustering-based superpixel algorithms where spatial information is used for distance computations, the possibility that a cluster centroid is the nearest cluster centroid of an input pixel is high if the cluster centroid and the input pixel are spatially close. Therefore, by setting priority for cluster centroids in the search range, the nearest cluster centroids of the input pixel can be found early. As shown in Fig. 1, the numbers in the blocks show the priority of the corresponding cluster centroids. The smaller the number, the higher the priority, which means that the cluster centroids that are spatially close to the input pixel are more likely to be the nearest cluster centroid. This feature is the key concept of this work, which is used to reduce distance computations.

The features and contributions of this work are stated as follows. Firstly, the spatial distances and the color distances are calculated individually, so that the redundant distance computations can be saved. Secondly, by searching the nearest
cluster centroids with centroid priority, the nearest clusters can be found at an early stage. Thirdly, the early-termination cluster centroids with centroid priority, the nearest clusters individual distance computations.

Fig. 2. Flowchart of the proposed clustering-based superpixel algorithms with individual distance computations.

**II. PROPOSED SUPERPIXEL ALGORITHM**

The left side of Fig. 2 shows the proposed superpixel algorithm, which is modified from the K-Means clustering algorithm [3]. The \( i \)-th 5-D input vector, which is extracted from the \( i \)-th pixel, is shown in (1).

\[
X_i = (X_{i,1}, X_{i,2}, X_{i,3}, X_{i,4}, X_{i,5}),
\]

where \((X_{i,1}, X_{i,2})\) is the 2-D spatial feature vector, which represents the vertical and the horizontal coordinates of the pixel, and \((X_{i,3}, X_{i,4}, X_{i,5})\) is the 3-D color feature vector, which can be extracted from any color spaces. The steps of the algorithm are described as follows.

- **Initialization of Centroids**: As shown in the right side of Fig. 1, partition the input image into small blocks with equivalent sizes \((S \times S)\) pixels, and initialize the cluster centroids by choosing one pixel from each block of the image. Assign the indices of blocks to the clusters, so that the access sequence of cluster centroids can be controlled. The vector of the \( j \)-th cluster centroid is the vector of the chosen pixel in the corresponding block, which is shown in (2).

\[
C_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}),
\]

where \((C_{j,1}, C_{j,2})\) is the 2-D spatial feature vector, which represents the vertical and the horizontal coordinate of the pixel, and \((C_{j,3}, C_{j,4}, C_{j,5})\) is the 3-D color feature vector.

- **Start of Iteration**: Decide the maximum number of iterations, and start the loop of iteration.
- **Start of Input Data Loop**: Process the input vectors \( X_i \) sequentially.
- **Initialization of Minimum Distance**: Set the minimum total distance to an initial value, which is a large number. Obtain the search range of clusters according to the spatial feature vector of the input vector \( X_i \). Fig. 1 shows an example of the search range, which contains \( 5 \times 5 \) cluster centroids. The center of the search range is the cluster centroid which corresponds to the image block containing the input pixel. The search range will not change for each pixel after the cluster centroids are updated.
- **Search of Nearest Centroids**: Search the nearest cluster centroid of the input vector \( X_i \). Among all the cluster centroids in the search range, the total distance between the nearest cluster centroid and the input vector \( X_i \) is the shortest. The proposed speed-up schemes, which are explained in the following subsections, are applied in this step.
- **Assignment of Input Data**: Based on the search result, assign the input vector \( X_i \) to the nearest cluster centroids.
- **End of Input Data Loop**: After all the input vectors \( X_i \) are processed, the loop ends.
- **Update of Centroids**: Update the vector of each cluster centroids \( C_j \) with the average vector of the input vectors which are assigned to it.
- **End of Iteration**: If the maximum number of iteration is reached, or the clustering result converges, the iteration stops. The clustering process ends, and the superpixels are dumped out. According to the requirement of applications, the spatial connectivity can be enforced with connected component labeling algorithms [15] [16].

**III. ACCELERATION SCHEMES**

The proposed 3 speed-up schemes, individual distance computations, centroid priority, and early termination, are applied in the “Search of Nearest Centroid” step in Fig. 2. The details are explained in the following subsections.

A. **Individual Distance Computations**

The flowchart of individual distance computations is shown in the right side of Fig. 2, and the steps are described as follows.

- **Start of Centroid Loop**: Process the vectors of cluster centroids \( C_j \) sequentially.
- **Computations of Spatial Distance**: Compute the spatial distance \( D_s \) of the input vector \( X_i \) and the vector of cluster centroid \( C_j \) based on the Euclidean distance metric, which is shown in (3).

\[
D_s(X_i, C_j) = \sum_{k=1}^{2} (X_{i,k} - C_{j,k})^2.
\]

If the spatial distance is larger than the minimum total distance, the computations of the total distance, update
of the minimum total distance, and the nearest cluster centroids in the next 2 steps are skipped. Otherwise, proceed to the next step.

- **Computations of Total Distance:** The color distance $D_c$ of the input vector $X_i$ and the vector of cluster centroid $C_j$ is computed based on the Euclidean distance metric, which is shown in (3).

$$D_c(X_i, C_j) = \sum_{k=3}^{5} (X_{i,k} - C_{j,k})^2.$$  \hspace{1cm} (4)

The total distance $D$ of the input vector $X_i$ and the vector of cluster centroid $C_j$ is the linear combination of the color distance $D_c$ and the spatial distance $D_s$, which is shown in (3).

$$D(X_i, C_j) = \omega \cdot D_s(X_i, C_j) + D_c(X_i, C_j),$$  \hspace{1cm} (5)

where $\omega = (m/S)^2$ is the weight of the spatial distance. $m$ is the variable used to adjust the compactness [13] of the region, and $S$ is the width (height) of the image block mentioned in Sec. II. If the total distance is larger than the minimum total distance, update of the minimum total distance and the nearest cluster centroid in the next step is skipped. Otherwise, proceed to the next step.

- **Update of Minimum Distance and Nearest Centroid:** Update the minimum total distance, and replace the nearest cluster centroid with the cluster centroid $C_j$.
- **End of Centroid Loop:** When all the vectors of cluster centroids $C_j$ in the search range are processed, terminate the centroid loop.

**B. Centroid Priority**

Fig. 1 shows an illustration of centroid priority. The smaller the number, the higher the priority. An example of applying centroid priority is shown in Fig. 3. Since the indices of cluster centroids are linked with the corresponding image blocks in the initialization phase, the access sequence and the search range can be controlled. The dot pattern and the line pattern represent computed distances and uncomputed spatial distances, respectively. The black pattern and the white pattern represent the minimum total distance and uncomputed color distances, respectively.

Fig. 3(a) shows the status before the search starts. The plot in the left side shows the relation between total distances and sequence numbers of cluster centroids, and the illustration in the right side shows the cluster centroids in the search range. It is an example of a $5 \times 5$ search range. The status after 9 cluster centroids are searched is shown in Fig. 3(b). The total distances between the input vector $X_i$ and the 4 cluster centroids (1st to 4th) are computed, and the 4th cluster is the nearest cluster. Since the spatial distances of the remaining 5 clusters are larger than the minimum total distance, it is not necessary to compute the color distances of the 5 clusters.

The status after 25 cluster centroids are searched is shown in Fig. 3(c). The total distances between the input vector $X_i$ and the 16 cluster centroids (10th to 25th) are computed, and the 4th cluster is still the nearest cluster. Since the spatial distances of the 16 clusters are larger than the minimum total distance, which is the total distance of the input vector and the 4th cluster, it is not necessary to compute the color distances of the 16 clusters.

By computing spatial distances and color distances individually, the redundant computations for color distances are saved, so that the computational speed is increased. Since the possibility that the nearest cluster centroid is spatially close to the input vector is high, the nearest cluster centroids can be found early with the centroid priority to reduce the computations of color distances.

**C. Early Termination**

The flowchart with early termination is shown in Fig. 4, and the step which is different from Fig. 2 is described as follows.

- **Computations of Spatial Distance:** Compute the spatial distance $D_s$ of the input vector $X_i$ and the vector of cluster centroid $C_j$ based on the Euclidean distance metric. If the spatial distance is larger than the minimum total distance and a threshold, skip the computations of the total distance, update of the minimum total distance, and the nearest cluster centroids in the next 2 steps; stop searching the nearest cluster centroid, and assign the input vector $X_i$ to the nearest cluster centroid. Otherwise, follow the steps in the right side of Fig. 2.

The threshold mentioned above can be either fixed or be dynamically adjusted. A method to automatically adjust the threshold within the search range is shown in (6).

$$T_{i,j} = \alpha \cdot \min_{1 \leq m < j} D(X_i, C_m),$$  \hspace{1cm} (6)

where $m$ indicates the sequence numbers of cluster centroids, and $\alpha$ is the weight of the minimum total distance.
The threshold $T_{i,j}$ changes according to the $i$-th input vector and the $j$-th cluster centroid. It is proportional to the current minimum total distance.

An example of early termination is shown in Fig. 5. The dot pattern and the line pattern represent computed distances and uncomputed spatial distances, respectively. The black pattern and the white pattern represents the minimum total distance and uncomputed color distances, respectively.

Fig. 5(a) shows the status before the search starts. The dot pattern and the line pattern represent computed distances and uncomputed spatial distances, respectively. The black pattern and the white pattern represents the minimum total distance and uncomputed color distances, respectively. The larger the size of the search range, the more computations can be skipped. When the size of the search range is larger than the minimum total distance and the threshold, it is not necessary to compute the color distances of the remaining 20 clusters.

Since the possibility that the nearest cluster centroid is spatially close to the input vector is high, the nearest centroids can be found early with the cluster centroid priority. If the spatial distance is larger than a threshold, the nearest cluster centroid might have already been found, so that the distance computations for all the remaining cluster centroids can be skipped. However, some errors might be incurred in the case that the computations for all remaining cluster centroids are skipped before the nearest cluster centroid is found. The smaller the value of the threshold, the higher the error rate.

IV. EXPERIMENTAL RESULTS

The Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500) [17] are used for the experiments. There are 500 images with 2,696 ground-truth images, and the size of images is $481 \times 321$ pixels. Fig. 8(a) shows a testing image in the BSDS500 database, and Fig. 8(b) shows one of the corresponding ground-truth images. The proposed speed-up algorithms are analyzed using 3 schemes. The 1st scheme includes only individual distance computations; the 2nd scheme includes individual distance computations and centroid priority; the 3rd scheme combines individual distance computations, centroid priority, and early termination. The experiments contain 4 parts. The 1st part is the analysis of computations, and the 2nd part shows the analysis of boundary recall and under-segmentation error. The 3rd part shows the comparison with related works [3] [13]. The number of superpixels before connectivity enforcement is 600 ($S = 10$), and the values of $m$ and $\alpha$ are set to 8 and 4, respectively. The centroids are initialized by choosing the pixel located in the center of each block. The color vectors are extracted from the RGB color space for the proposed algorithm and related works.

A. Analysis of Computations

The 1st part of the experiments is the analysis of computations. Fig. 6 shows the relation between the skipped computations and the size of search ranges. The “I” sign in the graph shows the standard deviation with the 500 images. When the size of the search range is $2 \times 2$, the skipped computations of scheme 1, scheme 2, and scheme 3 are 21.2%, 20.9%, and 23.9%, respectively; the standard deviations of scheme 1, scheme 2, and scheme 3 are 2.2%, 2.3%, and 2.9%, respectively. The larger the size of the search range, the more computations are skipped. When the size of the search range is $10 \times 10$, the skipped computations of scheme 1, scheme 2, and scheme 3 are 46.8%, 58.0%, and 95.4%, respectively; the standard deviations of scheme 1, scheme 2, and scheme 3 are 0.8%, 0.2%, and 0.5%, respectively. The results show that the standard deviations are small, which means that the performance does not vary significantly according to image contents, so that the proposed method can be applied to different kinds of images with similar computational costs.
section. When the size of the search range is
results after connectivity en forcement are discussed in this
ranges before and after connectivity enforcement [13]. The
relation between the boundary recall and the sizes of search
in scheme 3 because of early termination. Fig. 7(a) shows the
scheme 1 and scheme 2, but the accuracy might be influenced
segmentation. The same accuracy can be achieved using
performance metrics used to measure the accuracy of image
ary recall and under-segmentation error [18], which are the
B. Boundary Recall and Under-Segmentation Error
The 2nd part of the experiments is the analysis of bound-
ary recall and under-segmentation error [18], which are the
results after connectivity enforcement are discussed in this
section. When the size of the search range is $2 \times 2$, the values
of boundary recall of scheme 2 and scheme 3 are 71.87% and
71.93%, respectively; the standard deviations of scheme 2 and
scheme 3 are 8.26% and 8.23%, respectively. When the size of
the search range is $10 \times 10$, the values of distortion of scheme
2 and scheme 3 are 71.86% and 71.96%, respectively; the
standard deviations of scheme 2 and scheme 3 are 8.39% and
8.39%, respectively. The difference of scheme 2 and scheme 3
is very small in terms of boundary recall. Fig. 7(b) shows the
relation between the under-segmentation error and the sizes of
search ranges. When the size of the search range is $2 \times 2$, the values of under-segmentation error of scheme 2 and scheme 3 are both 20.10%; the standard deviations of scheme 2 and scheme 3 are 9.84% and 9.85%, respectively. When the size of
the search range is $10 \times 10$, the values of distortion of scheme
2 and scheme 3 are 20.15% and 20.14%, respectively; the
standard deviations of scheme 2 and scheme 3 are 9.84% and 9.85%, respectively. It is shown that the difference of scheme
2 and scheme 3 is very small in terms of boundary recall and under-segmentation error. The accuracy of scheme 3 is
almost the same as scheme 2, so that scheme 3 can reduce
computational costs without decreasing the accuracy.

C. Comparison with Related Works
The 3rd part of the experiments is the comparison with re-
lated work [3] [13]. Table I shows the summary of comparison,
and $N$ is the number of input pixels. In the K-Means clustering
algorithm [3], $K \times N$ operations are required for distance
computations. In our experiments, $K$ is equal to the number
of superpixels before connectivity enforcement, so that 600$N$
operations are required. It means that 600 cluster centroids
are searched for each pixel. In the SLIC algorithm [13], the

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Computations per Iteration</td>
<td>$K \times N$</td>
<td>$4N$</td>
<td>$(3.0 \pm 0.1)N$</td>
</tr>
<tr>
<td>Boundary Recall (%) [18]</td>
<td>71.86 ± 8.44</td>
<td>71.86 ± 8.38</td>
<td>71.94 ± 8.23</td>
</tr>
</tbody>
</table>

Storage of Label/Distance Images
None $\sim N$ None

Fig. 6. Analysis of skipped computations with scheme 1, 2, and 3.

Fig. 7. Analysis of (a) boundary recall and (b) under-segmentation error with scheme 2 and 3.

The 2nd part of the experiments is the analysis of bound-
ary recall and under-segmentation error [18], which are the
performance metrics used to measure the accuracy of image
segmentation. The same accuracy can be achieved using
scheme 1 and scheme 2, but the accuracy might be influenced
in scheme 3 because of early termination. Fig. 7(a) shows the
relation between the boundary recall and the sizes of search
ranges before and after connectivity enforcement [13]. The

TABLE I
COMPARISON WITH RELATED WORKS
distance images, whose size is proportional to the number of nearest centroids are searched in a $2S \times 2S$ local range, so that $4N$ operations are required for distance computations. It means that 4 cluster centroids are searched for each pixel, so that the SLIC algorithm is compared with the proposed algorithm with a search range of $2 \times 2$ centroids. In the proposed algorithm (scheme 3), only $3N$ operations are required for distance computations by applying early termination, so that 25% of distance computations can be saved. It means that only 3 cluster centroids are searched for each pixel in average. In addition, the proposed algorithm achieves low under-segmentation error and high boundary recall compared with related works. The clustering quality (distortion) can also be adjusted by changing the size of search range.

Unlike the SLIC algorithm [13], the proposed method does not require additional memory storage for label images and distance images, whose size is proportional to the number of input pixels $N$. When the bit lengths of the distance image and the label image are 16 bits and 32 bits, respectively, the storage for one image (including 24-bit pixels) is 1.3MB for the SLIC algorithm. For the proposed algorithm, the required memory size for image data is only 33% of the SLIC algorithm. Therefore, the proposed method can achieve the same performance as the related work with low computational costs and small memory sizes. Fig. 8(c)(d) show an example of superpixels generated by the SLIC algorithm and the proposed algorithm.

V. CONCLUSION AND FUTURE WORK

In this paper, a series of acceleration schemes for clustering-based superpixels algorithm is proposed. The contributions of this work are stated as follows. Firstly, the spatial distances and the color distances are calculated separately, so that the redundant distance computations can be saved. Secondly, by searching the nearest clusters in a spiral order, the nearest clusters can be found at an early stage. Thirdly, the early-termination mechanism can be applied to the search process to speed up the algorithm without decreasing the image segmentation quality. The experiments show that the proposed method achieves the level of performance as the SLIC superpixel algorithm with only 75% of distance computations and 33% of memory costs. The proposed acceleration schemes provide a tradeoff between the performance and the computational speed. Related applications and extensions of the proposed work will be discussed in our future work.

REFERENCES