applied sciences

MDPI

Article

NICE: Superpixel Segmentation Using Non-Iterative Clustering with Efficiency

Cheng Li [1], Baolong Guo [1,*], Geng Wang [1], Yan Zheng [1], Yang Liu [2] and Wangpeng He [1]

[1] Institute of Intelligent Control and Image Engineering, Xidian University, Xi'an 710071, Shaanxi, China; licheng812@stu.xidian.edu.cn (C.L.); wanggeng1996@stu.xidian.edu.cn (G.W.); yanzheng@stu.xidian.edu.cn (Y.Z.); hewp@xidian.edu.cn (W.H.)

[2] GLI Technology Limited, Shenzhen 518057, Guangdong, China; yliu@glitech.com

* Correspondence: blguo@xidian.edu.cn; Tel.: +86-180-9280-1271

Received: 1 June 2020; Accepted: 24 June 2020; Published: 26 June 2020

Abstract: Superpixels intuitively over-segment an image into small compact regions with homogeneity. Owing to its outstanding performance on region description, superpixels have been widely used in various computer vision tasks as the substitution for pixels. Therefore, efficient algorithms for generating superpixels are still important for advanced visual tasks. In this work, two strategies are presented on conventional simple non-iterative clustering (SNIC) framework, aiming to improve the computational efficiency as well as segmentation performance. Firstly, inter-pixel correlation is introduced to eliminate the redundant inspection of neighboring elements. In addition, it strengthens the color identity in complicated texture regions, thus providing a desirable trade-off between runtime and accuracy. As a result, superpixel centroids are evolved more efficiently and accurately. For further accelerating the framework, a recursive batch processing strategy is proposed to eliminate unnecessary sorting operations. Therefore, a large number of neighboring elements can be assigned directly. Finally, the two strategies result in a novel synergetic non-iterative clustering with efficiency (NICE) method based on SNIC. Experimental results verify that it works 40% faster than conventional framework, while generating comparable superpixels for several quantitative metrics—sometimes even better.

Keywords: superpixel segmentation; acceleration; inter-pixel correlation; recursive processing

1. Introduction

Superpixels perceptually group similar pixels into region-level features while heavily reducing the number of entities. Owing to the outstanding performance on region description, superpixel segmentation gradually becomes a fundamental pre-processing step in advanced visual tasks. Increasingly, many practical visual applications, such as image and video segmentation [1,2], target tracking [3], saliency detection [4] and remote sensing classification [5], are developed based on superpixels instead of pixel-wise features to achieve better performance.

Known as image over-segmentation [6], the target of superpixel segmentation is to generate a coherent grouping of pixels [7]. A growing number of superpixel algorithms have been put forward to improve the representative efficiency over the past two decades. Some prominent methods are incorporated into the state-of-the-art in this field. Meanwhile, several properties are commonly expected for good superpixels, such as accuracy, uniformity, compactness and efficiency [8]. Among them is Simple Linear Iterative Clustering (SLIC) [9], which typically provides an eligibly balanced trade-off between appearance homogeneity and shape regularity. Thanks to its concise yet enlightening principle, SLIC becomes more appropriate for deployment and expansion than other approaches in recent works [10–15]. Whereas in fact, deficiencies and shortcomings are also exposed in SLIC and its

variants for their iterative clustering framework. Firstly, redundant computations in overlapping local regions are repeated in more than an iteration. In addition, a split-and-merge heuristic is necessary for region connectivity [16]. What's worse, assignment and update steps within these methods are performed separately, leading to detrimental convergence.

Theoretically, SLIC is an instructive approach that restricts k-means algorithm into small windows to simplify the calculation of Centroidal Voronoi Tessellation (CVT) [17] in a 5-dimensional Euclidean space. Combining 3-dimensional digital value and 2-dimensional space features as weighted vectors, all pixels are repeatedly assigned to the nearest seeds till they all satisfy the convergence condition. Similarly, Linear Spectral Clustering (LSC) [12] introduces weighted k-means into a 10-dimensional feature space based on an elaborate kernel matrix. By preserving the global image properties, LSC achieves more perceptually satisfactory segmentation results. Zhao et al., proposed Fast Linear Iterative Clustering (FLIC) [15], in which a novel active search mechanism and a back-and-forth traversal strategy based on neighboring continuity is applied. It takes the place of fixed search ranges in other variants of SLIC and realizes rapid convergence of linear clustering. Intrinsic manifold SLIC (IMSLIC) [16] extends SLIC by an elaborate distance measurement and a 2-dimensional manifold feature space mapping. The new framework can make superpixels sensitive to image content without post-processing to enforce connectivity. In [10], the deviation of cluster centers in each iteration is adopted as a homogeneity cue to guide the convergence of candidate regions. Wherein the updating step, only some instable SLIC superpixels are updated by the local k-means method. Therefore, much redundant searching computation can be avoided. Superpixel Sampling Network (SSN) [11] proposes a new differentiable model for SLIC that turns it into a differentiable algorithm by relaxing the nearest neighbor constraints. In this way, conventional hand-crafted features can be trained by deep networks, which shows a better development prospect for this field.

More recently, to overcome the underlying limitations induced from the iterative k-means clustering within SLIC-like methods, Achanta et al., proposed simple non-iterative clustering (SNIC) [8] algorithm as an optimization. As the name suggests, SNIC performs in a non-iterative way and removes the limitations while it achieves desirable performance. However, SNIC overly relies on the priority queue to achieve non-iterative optimization. In fact, that is a greedy strategy and is prone to get trapped in a premature convergence problem than SLIC [18]. In addition, even if SNIC can be performed with high efficiency, its theoretical complexity is not always lower than SLIC [19]. On the other hand, SNIC adopts a rigid region growing method to generate superpixels, in which clustering centroids are evolved using online averaging. Similar to some SLIC-like methods, SNIC may suffer from the color-spatial compactness that goes against the local homogeneity, especially in complicated texture and low contrast regions.

To fulfill the aforementioned requirements, this paper proposes a new superpixel segmentation algorithm, referred to as non-iterative clustering with efficiency (NICE). It improves the conventional SNIC in two aspects. Firstly, a new strategy termed Elimination of Inspection Redundancy (EIR) is proposed. It introduces inter-pixel correlation as a local smooth cue to determine the priority for pixels with similar appearance. Therefore, large amounts of repetitions during the inspection process are avoided. By calculating neighboring information in this manner, weak boundaries between low contrast regions could be well preserved. In addition, an Accelerated Implementation based on Recursion (AIR) partially re-organizes the original SNIC. A last-in-first-out (LIFO) stack is introduced to speed up sorting local pixels in the priority queue. During the joint online assignment and updating step, a large number of pixels are processed in a batch, which gets rid of frequent push and pop operations, thus it avoids repeated pixel inspecting and sorting. Eventually, NICE combines EIR and AIR into SNIC framework in a subtle way, which inherits the desirable properties of original SNIC and the property of the two strategies.

To the best knowledge of the authors, this work is the first trial to optimize the SNIC algorithm. As its name indicates, the proposed NICE generates comparable superpixels with respect to boundary recall, under-segmentation error and achievable segmentation accuracy. It is more efficient and achieves

approximately 24 fps for a 481 × 321 image that exceeds SNIC's 16 fps without any parallelization or GPU processing.

This paper is organized as follows. The conventional SNIC framework is reviewed in the next section. In Section 3, the proposed two strategies EIR and AIR, as well as the NICE framework are presented in detail, respectively. Qualitative and quantitative analyses are explicated in Section 4. Finally, Section 5 makes a brief conclusion.

2. Simple Non-Iterative Clustering

Since the proposed improvements mainly works on SNIC, the conventional principle is reviewed at the beginning, as well as some notations and definitions in this paper. The basic idea of SNIC is demonstrated as follows:

- An input 3-channel Lab image $\{I_i\}_{i=1}^{N}$ is uniformly partitioned by a set of evenly distributed seeds $\{S_k\}_{k=1}^{K}$, where I_i represents the i th pixel in the image plane with N elements, S_k means k th centroid of mass in grids with a step of $s = \sqrt{N/K}$ and K is user-specified to expect the number of superpixels;
- For a pixel I_i in the image plane, it can be represented as a 5-dimensional Euclidean feature vector $E(I_i) = [l(I_i), a(I_i), b(I_i), x(I_i), y(I_i)]$. Specifically, $E(I_i)$ is composed of a vector of the 3-channel Lab digital values $C(I_i) = [l(I_i), a(I_i), b(I_i)]$ and image position coordinates $P(I_i) = [x(I_i), y(I_i)]$;
- In the initialization step each element of $\{S_k\}_{k=1}^{K}$ with unique labels are initialized on the uniform grid in the image plane as original cluster centers. A priority queue Q with increasing order is introduced which always returns the element $Q_{\min}$ with the minimum key value while it is not empty. For each element $I_i \in \{I_i\}_{i=1}^{N}$, $Q(I_i)$ is adopted to represent the distance to corresponding cluster center, and then recorded as the key value for sorting in Q. Specially, for each seed S_k, all information is included in a vector node $[E(S_k), k, Q(S_k)]$ with $Q(S_k) = 0$. Then all seed vectors are pushed on Q.
- In the joint assignment and updating step, $\{S_k\}_{k=1}^{K}$ is updated using online averaging of all clusters. For an unlabeled neighboring pixel I_i, inspected by the currently popped pixel whose cluster centered at S_k, the distance $Q(I_i)$ is calculated by Equation (1) where it is identical to $D(I_i, S_k)$. Then I_i is pushed on Q.

$$D(I_i, S_k) = \sqrt{\|C(I_i) - C(S_k)\|_2^2 + \lambda^2 \|P(I_i) - P(S_k)\|_2^2}, \tag{1}$$

  where λ is the quotient of maximal $C(I_i)$ and $P(I_i)$ within this cluster to normalize color and spatial proximity, and $\| \|_2$ represents the Euclidean metric;
- Followed by all neighboring pixels pushed around the frontier pixel, secondly, $Q_{\min}$ is acquired by popping the top-most element from Q, which corresponds to a pixel I_m containing the global minimum distance in Equation (2)

$$I_m = \arg\min D(I_i, S_k),\ i \in [1, N], k \in [1, K], \tag{2}$$

  that is, $Q_{\min} = Q(I_m)$. Then a new label is assigned to I_m in accordance with its nearest cluster center, and I_m turns into the frontier of its cluster. Meanwhile, the feature vector of the cluster centroid is updated by

$$E\left(\widetilde{S_k}\right) = \sum\nolimits_{i \in \Omega_k} [C(I_i), P(I_i)] \ / \ |\Omega_k|, \tag{3}$$

  where Ω_k means the cluster which is centered at S_k, and $|\Omega_k|$ means the number of pixels in Ω_k.
- The latter two procedures are repeated till Q is empty.

For further illustrating the principle of NICE vividly, as well as explaining the distinction from SNIC, a dynamic segmentation procedure on one 10 × 10 polychrome map fragment is partitioned in Figure 1. To keep it simple, four-quadrant initialization is adopted in following schematic diagrams.

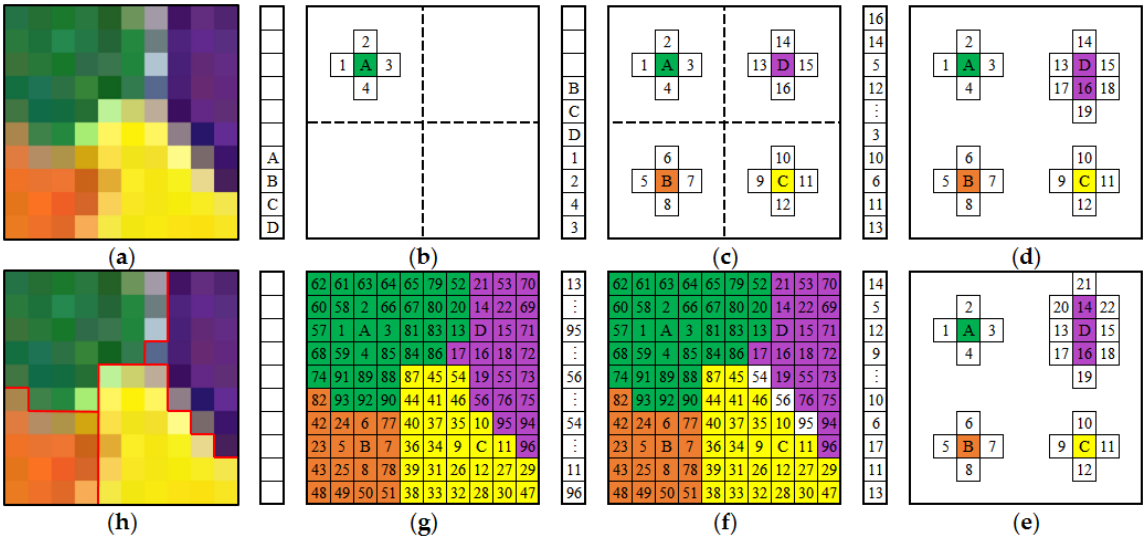

Figure 1. Dynamic segmentation procedure of simple non-iterative clustering (SNIC). (**a**) Input image; (**b**–**g**) execution of SNIC; (**h**) segmentation result. Among (**b**) to (**g**), each left column indicates the priority queue Q, in which the upmost element has the highest priority. The number in each pixel grid depicts the global order when it is inspected as a 4-neighbor element. Pixels with filled colors in label maps indicate that they have acquired labels corresponding to their nearest cluster centers. Note that the state of Q is one step earlier than the corresponding label map with pixel inspection and label assignment.

As shown in Figure 1b, four pixels are firstly pushed on the priority queue Q as initial seeds. Since they are identical to the original cluster centers, all seeds share the same priority. According to the order of entry, they are denoted by A, B, C and D from top to bottom, respectively. After that is popping the top-most element A, which becomes the current frontier pixel and then clockwise inspects its 4-neighbor elements.

Similarly, the other three seeds execute popping and inspecting successively. Notice that the seeds hold the priority, thus they still anticipate any neighboring pixels. As shown in Figure 1c, a preliminary label map is acquired after all seeds are popped. In addition, a total of 16 neighboring pixels are pushed and sorted in Q as candidates.

Figure 1d,e demonstrate the jointly repeated assignment and updates step. The neighboring elements 16th and 14th inspected by seed D are given as an example, respectively. After inspecting in Figure 1c, 16th becomes the top-most element in Q. It is then assigned with the label of seed D and treated as the frontier pixel of the cluster S_D, as well as absorbed to perform an online update of the centroid value. In Figure 1e, the newly inspected pixels, 17th, 18th and 19th are pushed before popping 14th elements. In terms of priority by the global minimum distance, 14th falls behind 16th but overtops any of the above latest pixels. Consequently, it performs a similar process as 16th. This kind of repeated procedure is continued until Q is empty.

Figure 1f,g are the state of Q (note that "before") as well as corresponding label map near and in the end, respectively. Eventually, the outlines of superpixels in Figure 1h are determined by the borders of pixel clusters with the same labels.

Since all superpixels are generated by seeds expanding that absorb surrounding pixels, SNIC mimics a geodesic distance from cluster centers to their corresponding frontier pixels. Therefore, the connectivity in the xOy coordinate space is guaranteed, and a post-processing step of split-and-merge is omitted.

3. Non-Iterative Clustering with Efficiency

This section introduces the proposed two strategies EIR and AIR, followed by the comprehensive NICE for superpixel segmentation. EIR evolves superpixel centroids in a more efficient and accurate way, which is based on inter-pixel correlation. In AIR, the new recursive implementation describes how to improve the sorting efficiency. At the end of this section, the overall NICE segmentation framework is summarized.

3.1. Elimination of Inspection Redundancy

In the procedure depicted in Figure 1, a large number of pixels are inspected several times since they become 4-neighbor elements of frontier pixels more than once. For example, in the upper left of Figure 2b, 62nd is newly inspected by 60th of the cluster that grows from B in green. Whereas it would be checked again in the next step due to 61st is the second-most prior element in that time. A different situation is boundary pixel such as 53rd in the upper right of Figure 2a, which is inspected by 21st, 22nd and 70th in different time. Particularly, in the orange regions of Figure 2c, 8th is inspected fully four times (B, 25th, 50th and 78th) from beginning to end. Generally, in conventional SNIC, a pixel in corners, boundaries and inner images is respectively inspected 2, 3 and 4 times, except it is labeled very early. Different from the first two situations, there are huge repetitions of inspection on the main internal pixels. It creates a large number of redundant elements in the priority queue that results in great computation and memory cost.

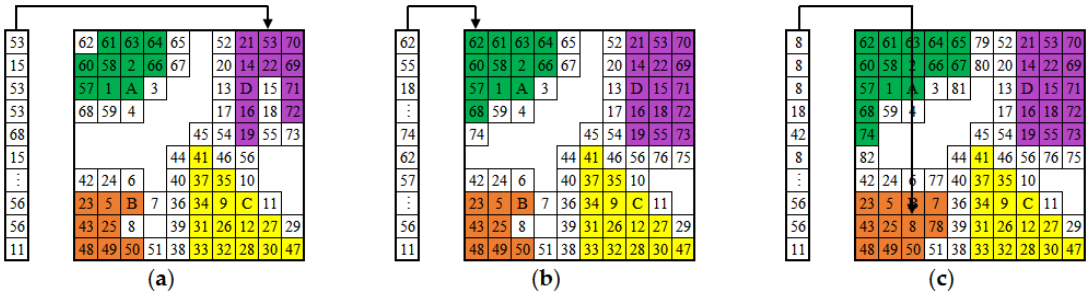

Figure 2. Local inspecting processes of SNIC in detail. Three representative pixels are shown as examples of inspection redundancy when they are labeled for the first time. (**a**) Boundary pixel 53rd; (**b**) corner pixel 62nd; (**c**) internal pixel 8th. Solid arrows stand for popping and labeling in corresponding label map. Notice that there are more than one identical elements with the top-most element in each priority queue.

Fundamentally, in the joint assignment and updating step of conventional SNIC, only the distance metric $D\left(I_j, S_k\right)$ in Equation (1) is adopted to sort priority of I_j in Q. It neglects the connection relationship of I_j and its corresponding frontier I_i. Afterwards, follow-up inspections on I_j may create other distance metrics as new elements of Q. When I_j becomes the top-most, it pops the minimum distance and other identical elements in Q are then meaningless for their corresponding clusters.

As a matter of fact, not the entire inspections on a neighboring element is necessary for its final attribution. In Figure 2, for example, based on the spatial label context, it is apparent by which cluster the isolated unlabeled pixels are eventually absorbed [20]. In other words, a newly inspected neighboring element would probably be assigned the same label as the corresponding frontier pixel, especially in some homogeneous regions. Thus, additional 1–3 inspections merely renew the minimum value of $Q\left(I_j\right)$ so that, more or less, it advances the assignment order of I_j in Q. Only in the controversial regions, such as the boundaries of objects in Figure 1f, multiple inspections may classify the pixels into clusters more reasonably.

Based on the above analysis, this subsection proposes a simple yet effective inspection strategy to eliminate the redundant creation of neighboring elements. The inter-pixel correlation between

adjacent pixels is taken into consideration, along with the distance relationship of pixel inspection and cluster center.

In what follows, the strategy referred to as Elimination of Inspection Redundancy (EIR) is described in more detail. Firstly in Figure 3a,b, the direction of 4-neighbor inspection is modified from clockwise rotation to cross where a frontier is the intersection. An intrinsic interaction on position is then enhanced for two pairs of partner pixels of unidirection (namely, left I_l to right I_r and up I_u to down I_d). As shown in Figure 3c, during the process of neighboring inspection on I_n, if I_l is newly checked and then labeled the same as I_n before I_r is assigned, I_r would be checked only once when

$$D_c(I_l, I_r) = \|C(I_l) - C(I_r)\|_2 < \sigma Q(I_n), \tag{4}$$

where the parameter σ is a threshold of color similarity. Here, D_c can be considered a digital value component of the distance metric in Equation (1), in which the difference in spatial distance can be omitted since the two elements are very spatially close. Similar calculations can be done for the pair of I_u to I_d. In other words, if the 4-neighbor elements of a frontier pixel satisfy Equation (4), two of them may be inspected in the current loop, and be omitted by other frontier pixels in the main clustering.

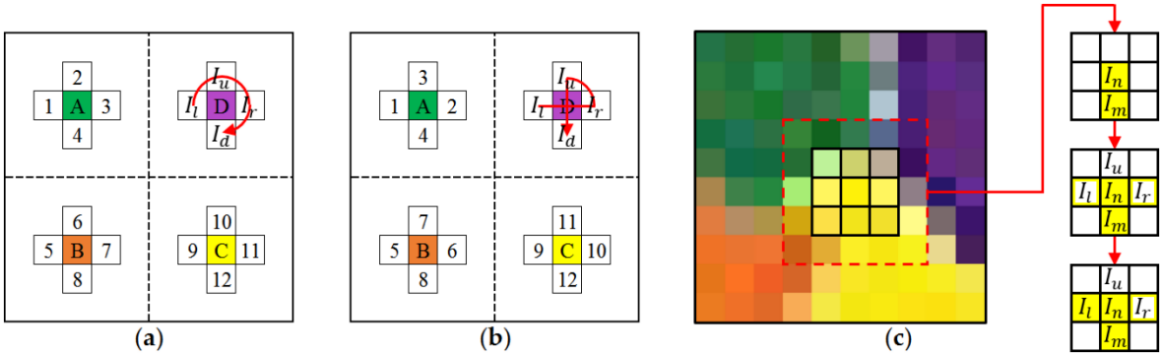

Figure 3. Elimination of Inspection Redundancy (EIR)-based inspecting and labeling processes of SNIC. (**a**) Inspection in clockwise rotation; (**b**) inspection in cross; (**c**) local inspecting process in a homogeneous region by EIR. Elements with the yellow box indicates that they are similar in color and satisfy Equation (4).

Theoretically in conventional SNIC, the similarity of each pixel with a cluster is determined by a joint digital value-spatial distance in Equation (1). It leads to unequal values for pixels that are in different positions due to the constraint of shape compactness, while they are in an identical color and adjacent positions. What is worse, it may accumulate the bias of a cluster that absorbs wrongly labeled pixels, causing a detrimental effect on center convergence. On the other hand, the proposed strategy strengthens the label correlation of pixels in the opposite position. It is based on the observation that if a neighboring pixel is associated with a certain cluster, its counterpart also highly tends to belong to that cluster. Therefore, it results in some boundary pixels merged by clusters that are more similar in the color feature, especially when they are in low contrast regions or complicated texture regions.

3.2. Accelerated Implementation Based on Recursion

The priority queue in SNIC is essentially a minimum heap, which is implemented by a complete binary tree in a logical structure and an array in physical storage. It adopts heapsort to sequence the elements that always returns the minimum value in the root node. A potential optimization is to improve the sorting efficiency of the priority queue in dynamic processes. Given a new inspected neighboring element, if its corresponding cluster distance is less than that of the frontier, it can be assigned directly without enqueueing and dequeueing. Thus, a subtle LIFO stack is embedded into the non-iterative clustering framework to achieve the acceleration in a recursive manner. This is

the so-called Accelerated Implementation based on Recursion (AIR). In addition, for an intuitive comparison of SNIC with and without AIR, the abovementioned EIR is not introduced in this subsection.

Figure 4 depicts the local inspecting and labeling processes from 48th to 50th in detail. Figure 5 illustrates the internal data structure updating of Q in Figure 4, aiming to explain the additional computation cost by redundant sorting. In a complete binary tree in Figure 5a, only the root node 48th is removed at a time, which is then replaced by the last leaf node 13th. Next, recursive adjustments occur in each subtree to keep the current roots minimum. The detailed operations are that a parent node interchanges with the smaller of its two child nodes till it is smaller than all leaf nodes. In Figure 5b, 49th is inspected after 48th is popped from Q, therefore it acts as the newly last leaf and update in a similar way to 13th. Since 49th takes precedence over 21st, it becomes the root of the tree temporarily as shown in Figure 5c. In the following step in Figure 5d, the tree adjusts itself in a similar manner as Figure 5a,b. These occurrences also suit for the upcoming 50th in Figure 5e,f. Notice that, 11th, 25th, 29th and 35th are duplicated respectively since they are checked twice. For simplicity, they are shown only once in the complete binary trees and the third time inspection on 25th between Figure 5d,e is omitted by 49th.

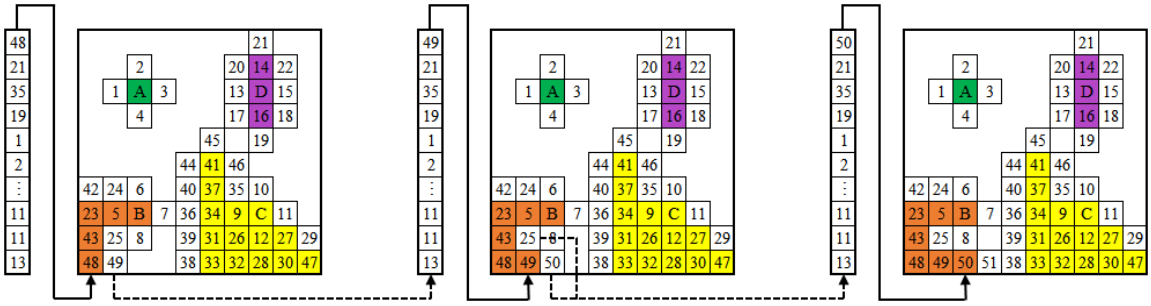

Figure 4. Detailed local inspecting and labeling processes from 48th to 50th sequentially of SNIC. Solid arrows stand for popping and labeling in corresponding label maps, while dotted arrows represent inspecting and pushing for the next step.

Comparing Figure 5a with Figure 5d, as well as Figure 5c with Figure 5f, two states of the binary tree are almost the same except for the roots. That is, for several pixels described in Figure 5, a number of data updating computations exist, which turn out to be redundant for sequential frontier pixels labeling and inspecting.

Aiming to the problem above, a last-in-first-out (LIFO) stack is introduced to pre-process the newly inspected pixels in advance, instead of pushing them on the priority queue directly. As shown in Figure 6, the priority queue is presented as an ordinary array, which is indeed a transposition of the column vector in Figure 4 before 43rd is popped. Unlike continuous queueing in Figure 4, this stack is embedded to the head of the array so that several pixels whose cluster distances are monotonic decreasing are processed in the batch.

Figure 6 demonstrates the array structure updating processes from 48th to 50th sequentially of SNIC. Specifically, in Figure 6b, 48th stays in the array head when 49th is inspected and pushed. After 49th acts as the current frontier pixel, it is superimposed on 48th and becomes the stack top. Then the newly inspected 25th and 50th perform a similar procedure to Figure 6b, while the former is filtered out by the LIFO stack and pushed on the priority queue. Finally, 48th to 50th in the stack are popped in batch. In the above process, only 25th and 48th are involved in global priority sorting after enqueueing, while the other two pixels derive from 48th and directly inherit its label without any additional computation.

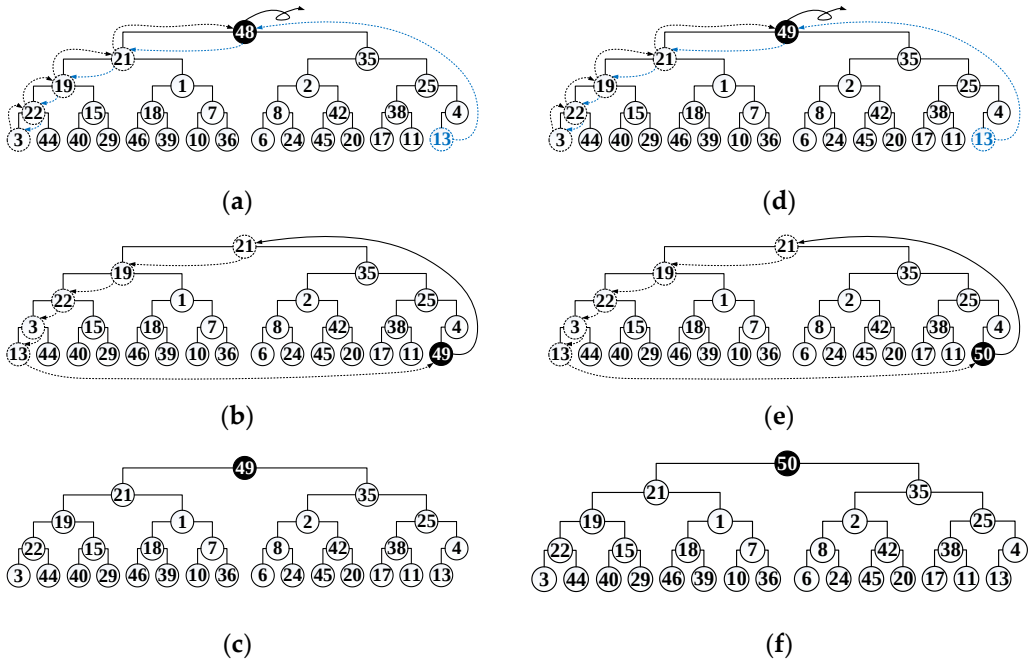

Figure 5. Heap structure updating processes of the priority queue from 48th to 50th sequentially in SNIC. (**a**) Popping 48th from Q; (**b**) pushing 49th on Q; (**c**) a temporary state after sorting 49th in Q; (**d**) Popping 49th from Q; (**e**) pushing 50th on Q; (**f**) a temporary state after sorting 50th in Q. Solid elements indicate that they are newly inspected and pushed in the current step, and their storage location is redirected by solid arrows. Other hollow nodes represent the elements that are inspected before but not labeled. Note that 13th is the last leaf node which recursively executes a sift-up or sift-down operation to modify the structure redirected by dotted arrows, as well as some of hollow nodes with dotted outlines.

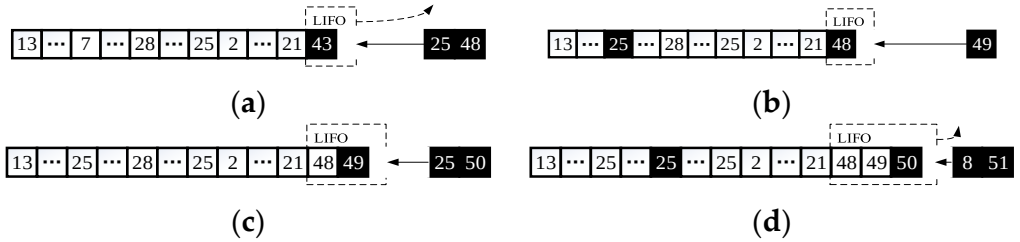

Figure 6. Array structure updating processes in the priority queue from 48th to 50th sequentially in SNIC. (**a**) Popping 43rd from Q before pushing 25th and 48th; (**b**) pushing 49th on Q; (**c**) pushing 25th and 49th on Q; (**d**) popping 48th to 50th from Q before pushing 8th and 51th. The sequence of elements is corresponding to the minimum heap in Figure 5. Solid elements indicate that they are newly inspected and pushed in the current and next step. A last-in-first-out (LIFO) stack with dotted arrows means that the elements are popped in the batch. Notice that there is more than one 25th in each array that is inspected by different pixels.

Therefore, the main idea of AIR can be briefed as follows. In general, for a pixel I_{new} inspected by one of its 4-neighbor elements that previously becomes the frontier pixel I_n, it can be directly assigned the same label k as the latter, only if

$$D(I_{new}, S'_k) \leq D(I_n, S_k), \tag{5}$$

where S_k and S'_k is the cluster before and after updated by merging I_n in Equation (3), respectively. The above equation can also be simply denoted as $Q(I_{new}) \leq Q(I_n)$. Recursively, for a subsequently inspected pixel I'_{new} derived from I_{new}, it can be directly assigned when $Q(I'_{new}) < Q(I_{new})$, wherein

the cluster is updated after merging I_{new}. The strategy also explains the reason why it fails to form a batch in Figure 6a ($Q(I_{25th}) > Q(I_{48th}) > Q(I_{43th})$) and 6d ($Q(I_{51th}) > Q(I_{8th}) > Q(I_{50th})$), respectively.

In addition to the single chained derivation demonstrated in Figure 4, a more complex situation is multi-pixels inspection with very similar priorities. Assuming that $Q(I_{8th}) < Q(I_{51st}) < Q(I_{50th})$ in Figure 6d (actually, the inequations in this paragraph are not satisfied), both two elements 8th and 51st fit Equation (5), they must be treated separately. Otherwise, if these two elements are together pushed onto the stack, some pixels supposed to be derived from 8th would be orphaned. Thus, only 8th with a higher priority is selected into the stack. Furthermore, since $Q(I_{51st}) < Q(I_{21st})$, for maintaining the logic of priority queue in conventional SNIC, 51st must be the new top-most element of Q before sorting the neighboring elements of 8th. That is, AIR must run in a single chained recursion that at most one neighboring element can be pushed on the stack. When the recursion originated from 8th is finished, 48th, 49th, 50th and 8th along with its derivations are all pushed in the batch with the label of 48th. Thus, the question turns into sorting in a priority queue that the top-most element is 51st. The above process runs in a recursive manner that the newly top-most element becomes the first element in the stack and inspects its neighbor in the next loop.

Another long-term example is 10 sequentially derived elements from 57th to 66th that originated from 1st. However in this process, only 4 of them (57th, 59th, 62nd and 65th) are pushed on the priority queue for sorting. For natural images, it is frequent to accumulate such a batch among adjacent pixels, especially in some homogeneous regions. It is worth noting, in theory, that the acceleration strategy of AIR maintains the global order of all pixels when they are inspected as 4-neighbor elements. Moreover, it adopts the intrinsic method of SNIC to assign the labels. Therefore, superpixels generated by the two methods would be consistent that only differ in computation cost. See Section 4 for qualitative and quantitative evaluations.

3.3. NICE Superpixel Segmentation Framework

A primary insight of the proposed NICE is to improve the efficiency of priority queue in dynamic processes, which can be generalized in two aspects. Firstly, a large number of repetitions during the inspection process is avoided by EIR. It introduces inter-pixel correlation as a local smooth cue to determine the priority for pixels with same digital values. Since the feature distance $Q(I_i)$ for sorting is no longer updated by other frontier pixels, the ultimate label of a neighboring element I_i is actually identified. Therefore, the assignment is just a matter of order. In addition, the non-iterative clustering framework is modified by AIR. It adopts an LIFO stack to the head of the priority queue and processes pixels in batch, thus meaningless enqueueing and dequeuing are reduced.

In practice, when σ is appropriate, early identified neighboring elements by EIR could promote AIR more outstanding. For example in Figure 6, if the relationship of 25th and 24th satisfies Equation (4) around 5th, there would be continuous queueing during this period, without sorting 25th twice. As a result, the combination of the above two strategies could significantly accelerate the non-iterative clustering framework. Moreover, the problem that digital value identical pixels are labeled differently by cluster updating and shape constraint is effectively alleviated, which achieves a desirable trade-off between runtime and accuracy. The integrated NICE superpixel segmentation framework is summarized in Algorithm 1.

Algorithm 1 NICE superpixel segmentation framework

Input: the Lab image I, the expected number K, the default normalization factor $\lambda = 10$
Output: the label map of $\boldsymbol{I}$
/* Initialization */
initialize cluster centers and assign starting labels similar to conventional SNIC.
initialize a priority queue Q with increasing order, and a LIFO stack S.
/* Joint assignment and updating */
for each cluster center S_k **do**
  create a vector node $[E(S_k), k, 0]$.
  push the node on Q.
end for
while Q is not empty **do**
  while S is not empty **do**
   pop the top-most node $[E(I_m), k, Q(I_m)]$ corresponding to I_m from S.
   goto AIR
  end while
  pop the top-most node $[E(I_m), k, Q(I_m)]$ corresponding to I_m from Q.
AIR:
  if I_m is not labeled before **then**
   assign the label of S_k to I_m.
   update the cluster S_k by Equation (3).
   set the global minimal pushed distance $Q_{\min} = Q(I_m)$.
   for all 4-neighboring unlabeled elements without EIR identification $I_n \in \{I_n\}$ of I_m **do**
    compute I'_n with the minimal pushed distance in $\{I_n\}$.
    compute the EIR correlation by Equation (4).
    if $Q(I'_n) < Q_{\min}$ **then**
     push the corresponding node $\left[E(I'_n), k, Q(I'_n)\right]$ on S.
     set the global minimal pushed distance $Q_{\min} = Q(I'_n)$.
    else
     push $\left[E(I'_n), k, Q(I'_n)\right]$ on Q.
    end if
    push the corresponding node $\left[E(I_n), k, Q(I_n)\right]$ on Q.
   end for
  end if
end while
return the label map of $\boldsymbol{I}$

4. Experiments and Analysis

Owing to its high performance, SNIC outperforms many other state-of-the-art methods such as [12] on execution time and accuracy [8]. Therefore, this section follows the experimental style in [20] that mainly focuses on the comparison of the proposed NICE and conventional SNIC. In addition, EIR-based Non-Iterative Clustering (ENIC) and AIR-based Non-Iterative Clustering (ANIC) that only adopt one strategy on SNIC are also implemented for comparison. The performance of the proposed approach is evaluated on the Berkeley Segmentation Data Set 500 (BSDS500) [21]. It contains 500 images with the size of 481×321 that are divided into three image sets, training (100), validation (200) and testing (200). The experiments are all evaluated on the test set using the benchmark toolbox proposed in [22] and implemented on an Intel Core i7-5500U laptop with a 2.4 GHz CPU.

4.1. Visual Comparisons of Superpixel Results

Figure 7 illustrates several subjective results for visual comparison of superpixels obtained by the above four algorithms. Benefiting from the outstanding performance of conventional SNIC, ANIC, ENIC and NICE all present relatively compact and uniform partitions. Theologically, the results of ANIC and SNIC, as well as ENIC and NICE are supposed to be the same, since AIR merely speeds up the sorting efficiency without any modification on clustering. Intuitively, it is hard to find a clear difference between them. Whereas some neighboring pixels with the same feature distance are pushed by a different order, the fluctuations in the convergence process may result in local bias on several superpixels (e.g., the worm tail). In addition, EIR alleviates the inevitably shape regulation in SNIC by color correlation. Therefore, both ENIC and NICE superpixels provide more accurate outline detection on complex object shape and weak boundaries.

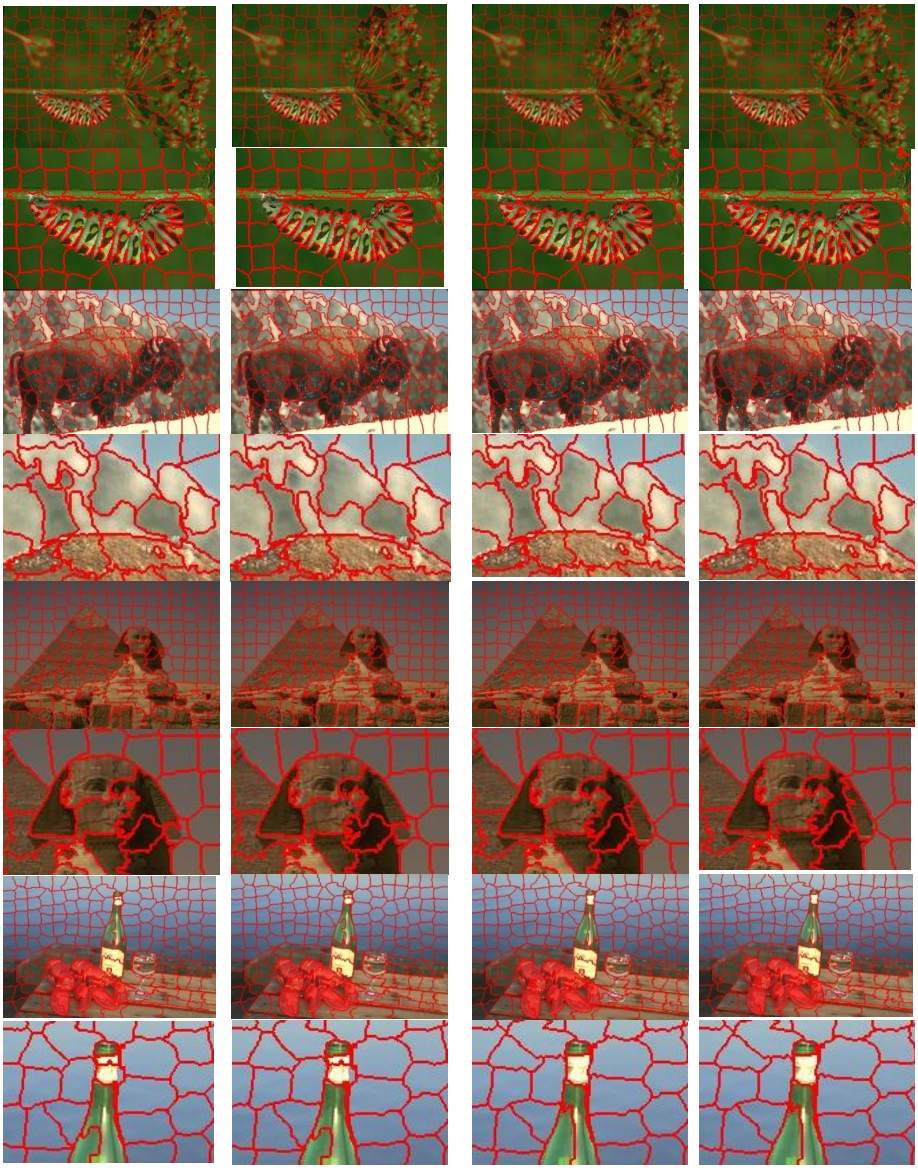

Figure 7. Visual comparison of superpixels with $K = 200$ on BSDS500. (**a**) SNIC; (**b**) Accelerated Implementation based on Recursion (AIR)-based Non-Iterative Clustering (ANIC); (**c**) EIR-based Non-Iterative Clustering (ENIC); (**d**) non-iterative clustering with efficiency (NICE). Alternating rows show each segmented image followed by local details of each image.

4.2. Quantitative Evaluation by Metrics

To objectively evaluate the performance of segmentation results, three evaluation metrics in [22] are taken into account, namely boundary recall (BR), under-segmentation error (UE) and achievable segmentation accuracy (ASA). All of them are commonly used in superpixel segmentation methods with emphasis on edge and region consistency, as well as the performance of subsequent visual tasks, respectively. More formally, let $\Omega = \{\Omega_k\}_{k=1}^{K}$ and $G = \{G_m\}_{m=1}^{M}$ be the calculated superpixels and the ground truth of the same image $\{I_i\}_{i=1}^{N}$, respectively. These metrics will be described in detail subsequently.

As mentioned above, there is little difference between SNIC and ANIC, as well as ENIC and NICE, thus the raw data of metrics in Tables 1–3 are more precise than figures (the curves are too close to distinguish visibly). The difference of values caused by AIR is marked in which blue and red indicate amelioration and deterioration, respectively. In the following tables and Figure 8, σ is set to 0.03 that merely verifies the effectiveness of EIR, and it would be covered in more detail in the next subsection.

Table 1. Comparison of SNIC and the proposed three algorithms in terms of boundary recall (BR).

Algorithm	User-Expected Number of Superpixels									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
SNIC	0.6740	0.7921	0.8428	0.8668	0.8985	0.9059	0.9195	0.9311	0.9387	0.9493
ANIC	0.6740	0.7922	0.8428	0.8669	0.8985	0.9060	0.9195	0.9311	0.9386	0.9493
ENIC	0.6757	0.7957	0.8457	0.8694	0.9010	0.9085	0.9213	0.9334	0.9405	0.9506
NICE	0.6757	0.7957	0.8457	0.8694	0.9010	0.9084	0.9212	0.9334	0.9405	0.9506

Table 2. Comparison of SNIC and the proposed three algorithms in terms of under-segmentation error (UE).

Algorithm	User-Expected Number of Superpixels									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
SNIC	0.1753	0.1100	0.0898	0.0809	0.0706	0.0680	0.0645	0.0603	0.0578	0.0545
ANIC	0.1751	0.1100	0.0897	0.0809	0.0706	0.0680	0.0645	0.0603	0.0578	0.0546
ENIC	0.1751	0.1089	0.0894	0.0806	0.0702	0.0675	0.0643	0.0603	0.0574	0.0544
NICE	0.1751	0.1088	0.0894	0.0807	0.0702	0.0675	0.0643	0.0603	0.0574	0.0544

Table 3. Comparison of SNIC and the proposed three algorithms in terms of achievable segmentation accuracy (ASA).

Algorithm	User-Expected Number of Superpixels									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
SNIC	0.8379	0.8962	0.9142	0.9222	0.9318	0.9344	0.9379	0.9413	0.9436	0.9464
ANIC	0.8379	0.8962	0.9142	0.9222	0.9318	0.9344	0.9379	0.9413	0.9436	0.9464
ENIC	0.8383	0.8966	0.9145	0.9225	0.9322	0.9347	0.9380	0.9416	0.9439	0.9465
NICE	0.8383	0.8967	0.9145	0.9225	0.9322	0.9347	0.9380	0.9416	0.9439	0.9465

Boundary recall (BR) is a standard boundary-detection and segmentation evaluation criterion. It can be described mathematically as the ratio of ground truth boundaries covered by superpixel boundaries [1]

$$\mathrm{BR} = \frac{\sum_{i \in G_b} \Pi\left(\min_{j \in \Omega_b} \|P(I_i) - P\left(I_j\right)\|_2 < r\right)}{G_b}, \tag{6}$$

where Ω_b and G_b represent boundary pixels in Ω and G, respectively. The indicator function $\Pi()$ returns the logic value whether the expression is true. In addition, the coverage radius r is set to 2 pixels in this paper. Therefore, the higher the value of BR, the better boundary adherence the algorithm performs. In Table 1, performance difference with and without AIR are ignorable, while ENIC and NICE benefits from EIR and outperform SNIC and ANIC, respectively. The reason is that

EIR introduces color context information to pixels that are adjacent to a common frontier inspector. Therefore, as depicted in Figure 3c, three pixels into a line with similar colors may be assigned an identical label. It efficiently alleviates the risk of shape constraint in complex boundaries. In addition, rather than AIR, EIR still plays a positive role in the non-iterative clustering framework, which also can be verified by the following two metrics.

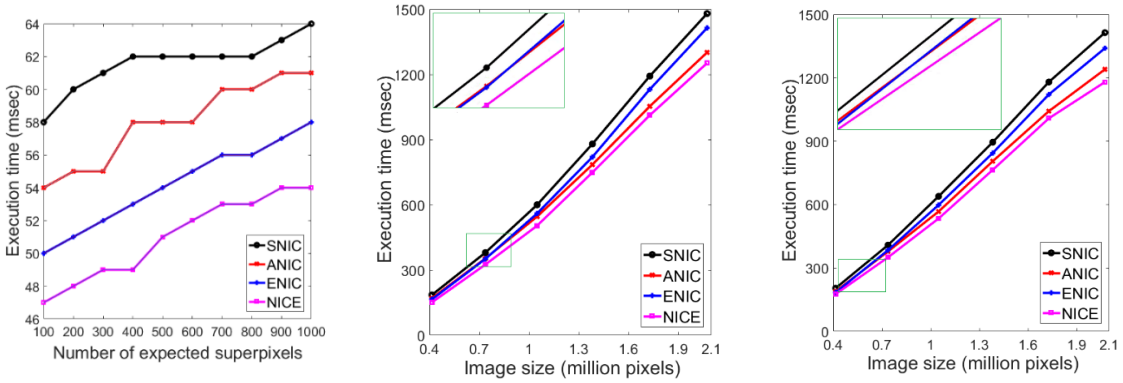

Figure 8. Comparison of runtime in milliseconds. (**a**) Time required for superpixels of increasing number in BSDS500; (**b**) time required for size-fixed superpixels of increasing size in multiple resolution image sets; (**c**) time required for number-fixed superpixels of increasing size in multiple resolution image sets. Green boxes show the local details of curves in (**b**,**c**).

Under-segmentation error (UE) measures how each superpixel overlaps with only one object

$$\mathrm{UE} = \frac{\sum_{m=1}^{M}\left(\sum_{\Omega_k|\Omega_k\cap G_m\neq\phi}|\Omega_k|\right) - N}{N}, \tag{7}$$

where || means the number of pixels in a superpixel. Compared with BR, it utilizes segmentation regions instead of boundaries for measurement. For large superpixels, theoretically, there is a serious penalty if they have only a small overlap with the ground truth segment. Thus in Table 2, as the number of superpixels grows, the values of UE for all four algorithms decrease. From another perspective, irregular superpixels tend to yield a higher UE since they may straddle over multiple object regions [1]. Therefore, the trade-off between runtime and accuracy by EIR can be accepted.

Achievable segmentation accuracy (ASA) is introduced to quantify the accuracy achievable by subsequent steps, such as image segmentation and object recognition. Mathematically, ASA can be computed by

$$\mathrm{ASA} = \frac{\sum_{k=1}^{K}\mathrm{argmax}_m|\Omega_k \cap G_m|}{\sum_{m=1}^{M}|G_m|}. \tag{8}$$

Similar to UE, it uses region information to evaluate the performance. A higher ASA value indicates the performance of superpixels in subsequent is unaffected. Similar to BR and UE, as a whole in Table 3, ASA becomes better along with the increasing number of superpixels. It also indicates that, in Figure 7, the visual difference caused by AIR mainly exists in the background, while the target contour is almost the same.

4.3. More Discussions on the Performance

The proposed NICE framework with two strategies allows conventional SNIC to be more efficient, therefore, some key factors that influence the performance are analyzed together. Figure 8 demonstrates

the comparison of execution time (ET) on four algorithms. The performance on the BSDS500 dataset with respect to different expected superpixel numbers is shown in Figure 8a. In addition, a number of supplementary natural images ranging from 720×576 to 1920×1080 with multiple characteristics, are packed into 6 sets to measure the efficiency in various image sizes. Figure 8b,c adopt a fixed size of superpixel (about 1600 pixels per region on average) and a fixed superpixel number (expect 1000 per image on average) to partition different size of images, respectively.

Theoretically, the time complexity of conventional SNIC is $O(N \log(n))$, wherein $\log(n)$ refers to the logarithm of the priority queue length. It means that the computational time is nearly linear in the number of pixels N in the image (as shown in Figure 8b). On the other hand, if there are a large number of unassigned elements in the priority queue, its sorting efficiency decreases significantly. For example, in Figure 8a, with the increasing expected number of superpixels, more unassigned neighboring pixels are inspected by additional seeds. It results in accumulation on the priority queue. It also can be proved by Figure 8b,c, where the difference of expected superpixel number is about 300 on 1920×1080 images.

As a parameter-free strategy in the non-iterative clustering framework, AIR exhibits a scale-sensitive property for accelerating. If there are a small amount of pixels in each superpixels on average, AIR fails to show a significant promotion on SNIC. On the contrary, as the size of input images grows, the average pixel number of each superpixels becomes greater, it gradually becomes the main contributor for NICE. On 1920×1080 images with about 1300 superpixels in Figure 8b, 24.2% of queueing operations are eliminated by AIR on average, which improves the efficiency by a 14% reduction in execution time.

Different from AIR, the result of EIR is determined by the parameter σ in Equation (4). As analyzed above, there is a balanced trade-off between runtime and accuracy by a proper σ. In other words, efficiency boosting is contradictory with other evaluation metrics. Since AIR scarcely affects the accuracy of ANIC and NICE, a more comprehensive discussion between SNIC and ENIC is expanded based on the value of σ.

Table 4 shows the difference in terms of several metrics under various values of σ on BSDS500 images with fixed superpixel number 200. Apparently, when $\sigma = 0$, ENIC is equivalent to SNIC. Once σ becomes positive, EIR starts to promote the non-iterative clustering framework and dramatically reduce the repetitions of inspection. On the other hand, it would worsen the performance due to an accumulation of digital value causing color bias σ grows larger. In this experiment, $\sigma = 0.08$ can be regarded as a balance for the abovementioned trade-off, which speeds up by 25% while preserves the accuracy of conventional SNIC.

Table 4. Differences of SNIC and ENIC in BR, UE, ASA and ET. "+/-" indicate the value of ENIC is greater/smaller than SNIC. Last row shows the ratio of neighboring inspection omitted by EIR.

Metrics	Parametric Value of σ									
	0.01	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**	**0.10**
BR	+0.0037	+0.0032	+0.0026	+0.0021	+0.0015	+0.0011	+0.0005	+0.0002	−0.0003	−0.0006
UE	−0.0003	−0.0003	-0.0003	−0.0002	−0.0001	−0.0001	0	0	0	+0.0001
ASA	+0.0003	+0.0003	+0.0003	+0.0002	+0.0001	+0.0002	+0.0001	+0.0001	+0.0002	+0.0002
ET (msec)	−7	−8	−9	−10	−11	−11	−12	−12	−13	−13
EIR ratio	+40.1%	+45.3%	+48.5%	+51.1%	+52.9%	+54.4%	+55.7%	+56.7%	+57.6%	+58.5%

It is notable that, for the resultant NICE, its speed-up ratio is greater than that sum of ANIC and ENIC. This indicates AIR together with EIR generates a synergetic effect that achieves more significant speed enhancement. Therefore, NICE only reduces the computation cost on small images, but lowers the slope of the required time for increasing image size. Given $\sigma = 0.08$ for EIR in Table 4, the execution time on BSDS500 images can be reduced to 42 milliseconds which is approximately 40% faster than SNIC.

To further verify the synergetic NICE, the performance is compared with SLIC, FLIC and SNIC on another benchmark dataset NYUv2 [23]. The dataset is converted by [7] for superpixel evaluation,

which contains 399 test images with size 608×448. For fair comparison, the default parameters of original code released online is used in other three algorithms, while σ is still set to 0.03 in NICE.

Figures 9 and 10 show the qualitative and quantitative performance of four methods on NYUv2, respectively. As shown in Figure 10, the iterative label updating methods SLIC and FLIC almost outperform the non-iterative label expansion methods SNIC and NICE on three metrics. Since the size of images of NYUv2 is larger than those in BSDS500, and the expected superpixel number K is limited in [50, 500], the average superpixel size in NYUv2 is greater than in BSDS500. Therefore, $2s \times 2s$ search region for each seed in SLIC provides more context information than SNIC, which easily gets into local optimum with a step $s \times s$ in grid initialization. Moreover, FLIC assumes that neighboring pixels have natural continuity, which is similar to inter-pixel correlation in EIR of NICE. Therefore, the active search strategy avoids the clusters being limited in fixed range in space, thus achieves more desirable boundary adherence. On the other hand, it results in less effect of compactness on controlling the regularity. As shown in Figure 9b, FLIC performs the worst in shape uniformity, which is too sensitive to maintain compact outlines among neighboring superpixels, even they all belongs to a homogeneous region (e.g., the floor and the wall). On the contrary, superpixels generated by SNIC and NICE are regular in both size and shape, while the former pays more attention to weak boundaries without apparent distortion. In addition, curves of metric on SLIC and FLIC tend to level off, whereas SNIC and NICE becomes better with the increasing of superpixel number.

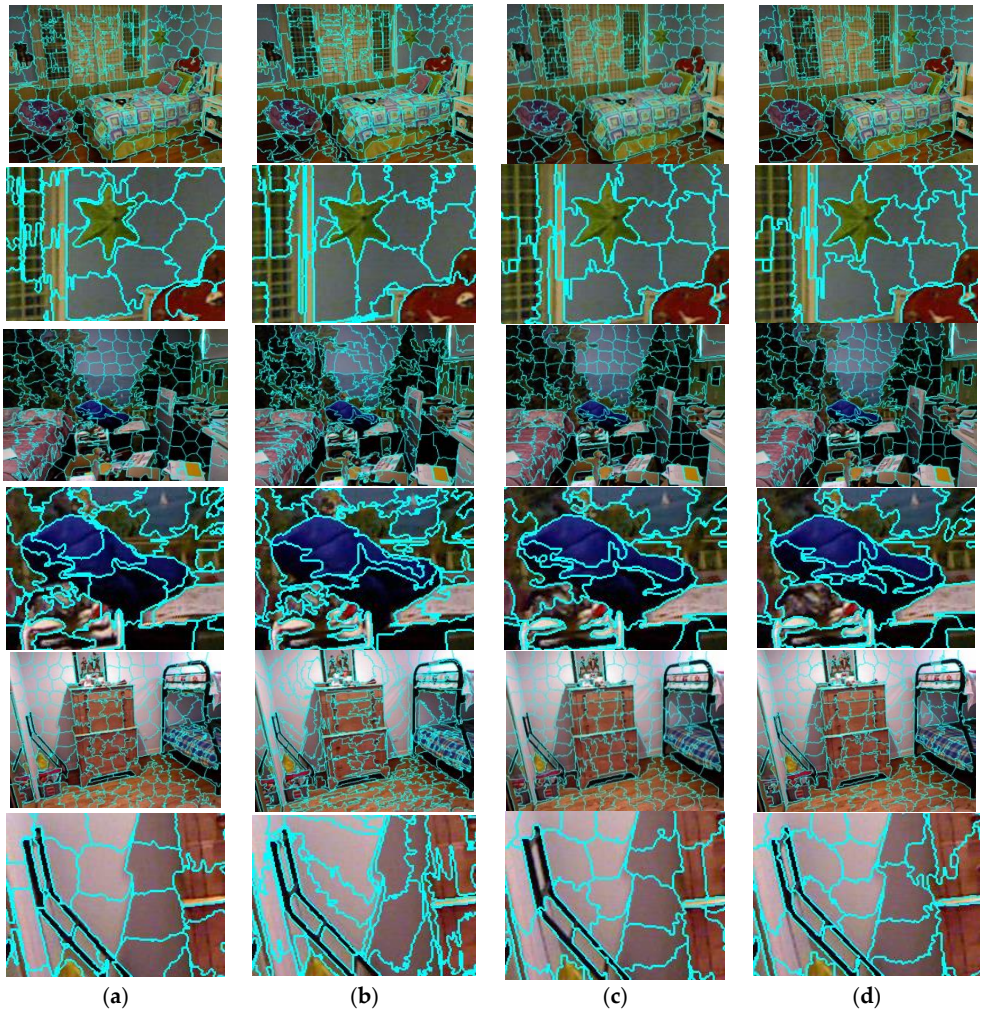

Figure 9. Visual comparison of superpixels with $K = 200$ on NYUv2. (**a**) SNIC; (**b**) FLIC; (**c**) SNIC; (**d**) NICE. Alternating columns show each segmented image followed by local details of each image.

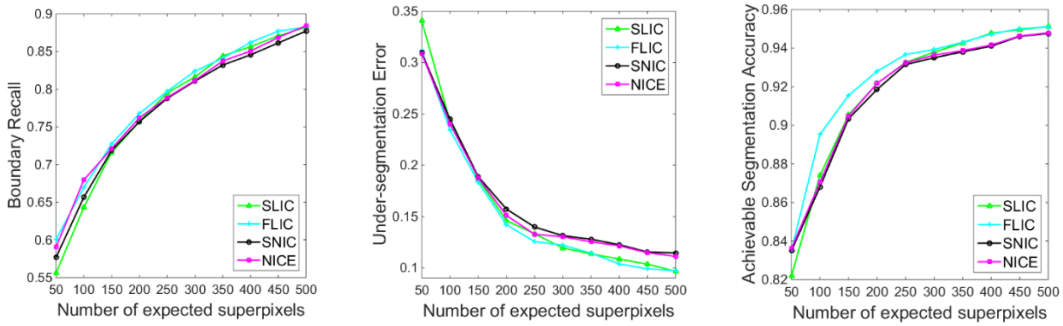

Figure 10. Performance comparison of four algorithms in terms of three quantitative metrics on NYUv2. (**a**) BR; (**b**) UE; (**c**) ASA.

Table 5 shows the execution time (ET) of four algorithms. Owing to the fast convergence by active search strategy, FLIC runs over 50% faster than SLIC and becomes one of the fastest state-of-the-art methods. As a synergetic effect of AIR and EIR, NICE maintains stable acceleration of SNIC on NYUv2 that exceeds FLIC more than 20%. Thus, it is accepted that the proposed NICE makes a reasonable trade-off between performance and efficiency.

Table 5. Comparison of four algorithms in terms of ET (msec) on NYUv2.

Algorithm	User-Expected Number of Superpixels									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
SLIC	123	124	124	124	126	126	128	129	130	130
FLIC	75	77	77	79	79	80	80	82	83	84
SNIC	69	71	73	75	76	76	77	80	80	81
NICE	62	63	64	65	66	66	67	68	69	70

5. Conclusions

In this paper, an accelerated non-iterative clustering framework that combines two strategies is proposed for an efficient and accurate generation algorithm of superpixels. The first strategy referred to as Elimination of Inspection Redundancy (EIR) deals with the redundant creation during neighbor inspection via the inter-pixel correlation between adjacent pixels. It also alleviates the inevitably shape regulation in complex boundaries thus efficiently generating desirable superpixels in both size and shape. The second strategy Accelerated Implementation based on Recursion (AIR) carries out the non-iterative clustering framework in a recursive manner. It introduces a subtle LIFO stack to directly assign the elements whose cluster distances are monotonic decreasing. Therefore, a large number of redundant sorting operations can be avoided without apparent deterioration of accuracy. A more efficient framework is integrated by EIR and AIR in a subtle way. The resultant NICE algorithm generates a synergetic effect and performs significantly better than conventional SNIC as well as the combination with any one of the strategies. Experimental results demonstrate that NICE is 40% faster than SNIC with comparable quantitative metrics on the BSDS500 dataset.

For further improving the performance of non-iterative clustering framework, more effective initialization methods are desired, which would overcome the limitation of grid sampling and cover more small objects and strip areas. It is also worth exploring size and compactness adaptive segmentation algorithms, which could conform to the varying content of images more reasonably.

Author Contributions: All the authors contributed to this study. C.L.: conceptualization, investigation and writing of the original draft; B.G.: funding acquisition and project administration; G.W. and Y.Z.: software and data curation; Y.L.: resources; W.H.: supervision and writing of review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported financially by National Natural Science Foundation of China (Grant No. 61571346 and 51805398).

Acknowledgments: The authors would like to thank the editor and anonymous reviewers for their valuable comments on this paper.

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

References

1. Xu, L.; Luo, B.; Pei, Z.; Qin, K. PFS: Particle-filter-based superpixel segmentation. *Symmetry* **2018**, *10*, 143. [CrossRef]
2. Boemer, F.; Ratner, E.; Lendasse, A. Parameter-free image segmentation with SLIC. *Neurocomputing* **2018**, *277*, 228–236. [CrossRef]
3. Yeo, D.; Son, J.; Han, B.; Han, J. Superpixel-based tracking-by-segmentation using Markov chains. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 511–520.
4. Liu, Z.; Zou, W.; Meur, O. Saliency tree: A novel saliency detection framework. *IEEE Trans. Image Process. (TIP)* **2014**, *23*, 1937–1952.
5. Zou, H.; Qin, X.; Zhou, S.; Ji, K. A likelihood-based SLIC superpixel algorithm for SAR images using generalized gamma distribution. *Sensors* **2016**, *16*, 1107. [CrossRef] [PubMed]
6. Baatz, M.; Schäpe, A. Multiresolution segmentation: An optimization approach for high quality multi-scale image segmentation. In Proceedings of the Symposium for Applied Geographic Information Processing, Karlsruhe, Germany, 5–7 July 2000; pp. 12–23.
7. Stutz, D.; Hermans, A.; Leibe, B. Superpixels: An evaluation of the state-of-the-art. *Comput. Vis. Image Underst.* **2018**, *166*, 1–27. [CrossRef]
8. Achanta, R.; Susstrunk, S. Superpixels and polygons using simple non-iterative clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4895–4904.
9. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Susstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
10. He, W.; Li, C.; Guo, Y.; Wei, Z.; Guo, B. A two-stage gradient ascent-based superpixel framework for adaptive segmentation. *Appl. Sci.* **2019**, *9*, 2421. [CrossRef]
11. Jampani, V.; Sun, D.; Liu, M.; Yang, M.; Kautz, J. Superpixel sampling networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 363–380.
12. Chen, J.; Li, Z.; Huang, B. Linear spectral clustering superpixel. *IEEE Trans. Image Process.* **2017**, *26*, 3317–3330. [CrossRef]
13. Giraud, R.; Ta, V.; Papadakis, N. Robust superpixels using color and contour features along linear path. *Comput. Vis. Image Underst.* **2018**, *170*, 1–13. [CrossRef]
14. Liu, Y.; Yu, C.; Yu, M.; He, Y. Manifold SLIC: A fast method to compute content-sensitive superpixels. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 651–659.
15. Zhao, J.; Hou, Q.; Ren, B.; Cheng, M.; Rosin, P. FLIC: Fast linear iterative clustering with active search. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New Orleans, LA, USA, 2–7 February 2018; pp. 7574–7581.
16. Liu, Y.; Yu, M.; Li, B.; He, Y. Intrinsic manifold SLIC: A simple and efficient method for computing content-sensitive superpixels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 653–666. [CrossRef]
17. Du, Q.; Faber, V.; Gunzburger, M. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Rev.* **1999**, *41*, 637–676. [CrossRef]
18. Wu, C.; Zhang, L.; Zhang, H.; Yan, H. Superpixels using fuzzy simple linear iterative clustering and fast precise number control. *arXiv* **2018**, arXiv:1812.10932.
19. Kang, X.; Zhu, L.; Ming, A. Dynamic random walk for superpixel segmentation. *IEEE Trans. Image Proce* **2020**, *29*, 3871–3884. [CrossRef] [PubMed]
20. Choi, K.; Oh, K. Subsampling-based acceleration of simple linear iterative clustering for superpixel segmentation. *Comput. Vis. Image Underst.* **2016**, *146*, 1–8. [CrossRef]

21. Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 898–916. [CrossRef]
22. Wang, M.; Liu, X.; Gao, Y.; Ma, X.; Soomro, N. Superpixel segmentation: A benchmark. *Signal Process. Image Commun.* **2017**, *56*, 28–39. [CrossRef]
23. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In Proceedings of the European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; pp. 746–760.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).