applied sciences

MDPI

Article

Image Inpainting Based on Multi-Patch Match with Adaptive Size

Shiyuan Yang, Haitao Liang, Yi Wang, Huaiyu Cai and Xiaodong Chen *

Key Laboratory of Opto-Electronics Information Technology of Ministry of Education, School of Precision Instrument & Opto-Electronics Engineering, Tianjin University, Tianjin 300072, China; yangshiyuan@tju.edu.cn (S.Y.); htliang@tju.edu.cn (H.L.); koala_wy@tju.edu.cn (Y.W.); hycai@tju.edu.cn (H.C.)

* Correspondence: xdchen@tju.edu.cn; Tel.: +86-22-2740-4535

Received: 10 June 2020; Accepted: 13 July 2020; Published: 17 July 2020

Featured Application: The work uses a patch-based inpainting method that is capable of filling the missing areas or removing unwanted objects in a digital image.

Abstract: Patch-based image inpainting methods iteratively fill the missing region via searching the best sample patch from the source region. However, most of the existing approaches basically use the fixed size of patch regardless of content features nearby, which may lead to inpainting defects. Also, global match is needed for searching the best sample patch, but only to fill one target patch in each iteration, resulting in low efficiency. To handle the issues above, we first evaluate the nonuniformity in an image, by which the patch size is adaptively determined. Moreover, we divide the source region into multiple non-overlapping subregions with different nonuniformity levels, and the patch match proceeds in every subregion, respectively. This strategy not only saves the match time for single target patch, but also reduces the mismatch, and enables the simultaneous filling of multiple target patches in a single iteration. Experimental results show that in comparison to previous patch-based works, our method has achieved further improvement both in quality and efficiency. We believe our method could provide a new way for patch match with better accuracy and efficiency in image inpainting tasks.

Keywords: image inpainting; nonuniformity; adaptive patch size; subregion search; multi-patch match

1. Introduction

Digital image inpainting is one of the research hotspots in the field of image restoration, which fills the missing areas with plausible content or replaces the unwanted objects with background utilizing the neighborhood information in digital images. Typical applications are such as restoration of damaged photos and ancient paintings, filling the holes in a virtual-view image [1], and removing the watermark or text in a picture. The purpose is to make the restored image seem as natural as possible, without noticeable traces of inpainting.

There are mainly two categories of traditional approaches for image inpainting: diffusion-based methods and patch-based methods. The general principle of the diffusion-based methods is to diffuse the known information into the missing regions in an iterative process, modeled by partial differential equation (PDE). Inspired by the propagation of heat flow, Bertalmio et al. [2] introduced the first diffusion-based method, and proposed the strategy of propagating the linear structure (i.e., isophote) from the source region (i.e., known region) into the missing region. Chan et al. [3] applied the total variational to the image inpainting for the first time (a.k.a. TV model), which converted the image inpainting into a mathematical problem that using the Euler-Lagrange equation to solve the extreme of

an energy functional established by the incomplete image. The TV model only relies on the gradient value of the isophote, rather than the geometric information, and always tends to use the shortest straight line to connect broken linear structures, which cannot satisfy the principle of visual connectivity. Later, based on TV model, Shen et al. [4] developed an improved curvature driven diffusion model (a.k.a. CDD model) to control the diffusion intensity, the CDD model has made up for the shortcomings of the TV model. Oliveira et al. [5] filtered the missing areas with Gaussian convolution kernel to diffuse the known pixel information into the unknown region.

Inpainting algorithms based on PDE could maintain the linear structure properly. However, these methods can only fill a very small number of pixels each iteration, and it still takes a long time even if to fill a small missing area. Although Gaussian-convolution based method is simple and fast, it fails to hold the linear structure. Generally, diffusion-based methods can only obtain better effect when the missing area is small, when it comes with a larger missing area, these methods may introduce visual blurring defects as the filling progresses, resulting in low inpainting quality.

In order to solve the problem that diffusion-based methods are more likely to introduce blurring defects when dealing with larger missing area, another category, the patch-based methods were proposed on the basis of the related research of texture synthesis and stitching [6–9]. The central idea is to select an appropriate sample texture patch from the source region to fill the unknown region under the certain rules. Because these algorithms use the texture patch as the filling unit rather than the single pixel, they could capture the local texture features better; therefore, patch-based methods can extend the linear structure without introducing blurring defects, and fill more pixels per iteration.

Generally speaking, patch-based methods outperform diffusion-based methods in both quality and efficiency, and are commonly used in image inpainting. One of the most representative pioneer works of patch-based methods is the exemplar-based image inpainting algorithm proposed by Criminisi et al. [10], which has also become the baseline work of other related patch-based methods. These methods mainly focus on two issues: One is to improve the filling order [11–14], the other is to find a better patch match criterion [15–17] (see Section 2 for details)—and these two issues are almost tackled. In addition to this, Zhou et al. [18] used dynamic patch size rather than the original fixed one to make the inpainting more flexible with different textures, but significantly increased the computational complexity. Moreover, all of the above methods (except [17]) still need to traverse the entire source region to find the best sample patch, and only to fill a single target patch in a single iteration, which requires lots of iterations and match time. In fact, low efficiency caused by iterative match process is known as a common problem for the patch-based methods. Although Liu et al. [17] tried to narrow the match area by picking out those candidates whose sum of the pixel values is close to the target patch's, this strategy is not an effective way to filter out those bad sample patches. Barnes et al. [19] limited the match area to the neighborhood of the target patch—this method is likely to miss the optimal patch if it is not located nearby. How to speed up the inpainting process while maintaining the quality remains a challenge.

Our work also uses a similar framework as Criminisi's [10], with several improvements that have successfully tackled the aforementioned issues. Our novel solution not only adopts the dynamic size of the patch to improve inpainting quality, but also narrows down the match area effectively and harmlessly, and reduces the total iterations by enabling multi-patch match strategy to achieve further inpainting efficiency. More specifically, our main novelties and contributions are:

(1) We first propose a metric to evaluate the nonuniformity in an image, and;
(2) To achieve a more accurate and flexible inpainting, the patch size is adaptively determined according to its nonuniformity;
(3) To save the match time, our subregion search strategy allows the match only between patches with similar content. This trick not only helps to narrow down the match area to a large extent while without missing the optimal sample patch, but also skips those bad sample patches to avoid mismatch;

(4) To reduce the total number of iterations, our multi-patch match strategy enables the patch match to proceed in multiple subregions with different nonuniformity levels, so that multiple target patches can be filled in a single iteration;

The rest of the paper is organized as follows: Section 2 will briefly introduce the classical Criminisi's algorithm, the baseline of our work. Section 3 shows the details of our improvements, including the nonuniformity model, determination of adaptive patch size, strategy of subregion search, and multi-patch match. Experimental results and analysis will be given in Section 4, and we compare our results with related patch-based methods proposed in recent years. Section 5 draws the conclusion and presents future work.

2. Related Work

We choose Criminisi's algorithm [10] as the baseline of our work since it is the pioneer work and has the most representative framework in patch-based inpainting methods. In this section, we first briefly introduce how it works, and then give some analysis about its shortcomings and the improvements in other related works.

For the convenience of expression, we define some notations first. As shown in Figure 1, I represents the entire incomplete image, the missing area (target region) is represented by Ω, Φ is the known area (source region) and is defined as $\Phi = I - \Omega$. $\delta\Omega$ is 1-pixel-wide outer boundary of Ω ($\delta\Omega \subset \Phi$), which is called the filling front, other symbols will be introduced later.

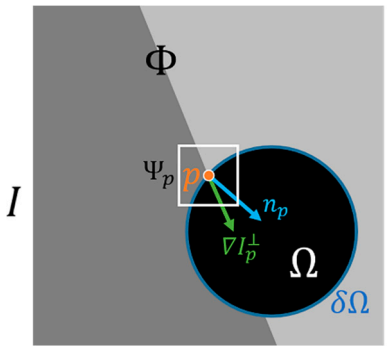

Figure 1. Notation diagram for Criminisi's algorithm.

The central idea of Criminisi's algorithm is that in each iteration, find the point with the highest priority on the filling front, and establish a target patch centered at the point, then globally search for the best sample patch; finally, the best sample patch is copied to the target patch to fill its unknown part. Repeat the above steps iteratively until the missing area is completely filled. Specifically, the algorithm consists of the following three steps:

1. Calculate the priorities along the filling front

In order to determine the filling order, the priorities need to be computed for all pixels along the filling front $\delta\Omega$. Given a pixel p ($p \in \delta\Omega$), the priority $P(p)$ is defined as follows:

$$P(p) = C(p)D(p) \ \forall p \in \delta\Omega, \tag{1}$$

where $C(p)$ is the confidence term and $D(p)$ is the data term, they are defined as follows:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \Phi} C(q)}{|\Psi_p|}, \tag{2}$$

$$D(p) = \frac{\left|\nabla I_p^{\perp} \cdot n_p\right|}{255}, \tag{3}$$

where Ψ_p is a 9×9 patch centered at p, and $|\Psi_p|$ is the area of Ψ_p,∇I_p represents the gradient at p and $\perp$ is the orthogonal operator, so $\nabla I_p^{\perp}$ denotes the isophote vector. n_p is a unit vector orthogonal to the filling front $\delta\Omega$ at the point p (see Figure 1). It can be seen that $C(p)$ represents the amount of known information contained in patch, while $D(p)$ shows how strong a linear structure contained in Ψ_p. The priority model encourages those areas with more known pixels and strong linear structures to be filled first. The initial value of $C(p)$ is set to $C(p) = 1$ $(\forall p \in \Phi)$, $C(p) = 0$ $(\forall p \in \Omega)$.

2. Search for the best sample patch to fill one target patch

After calculating the priorities of all pixels on the filling front $\delta\Omega$, find the pixel $\hat{p}$ with the highest priority, establish a target patch $\Psi_{\hat{p}}$ centered at $\hat{p}$, use a sample patch Ψ_q of the same size as $\Psi_{\hat{p}}$ to traverse the entire source region Φ to search the best sample patch $\Psi_{\hat{q}}$ that is most similar to $\Psi_{\hat{q}}$. The similarity metric is the sum of squared difference (SSD) between the known pixels in $\Psi_{\hat{p}}$ and corresponding pixels in $\Psi_{\hat{q}}$. The best sample patch $\Psi_{\hat{q}}$ satisfies the following equation:

$$\Psi_{\hat{q}} = \underset{\Psi_q}{argmin} \mathrm{SSD}(\Psi_{\hat{p}}, \Psi_q), \tag{4}$$

then the $\Psi_{\hat{q}}$ is copied to the unknown part of the $\Psi_{\hat{p}}$ so that one target patch is filled.

3. Update information

After $\Psi_{\hat{p}}$ is filled, the update rule of confidence term for new pixels p' in $\Psi_{\hat{p}}$ is as follows:

$$C(p') = C(\hat{p}) \ \forall p' \in \Psi_{\hat{p}} \cap \Omega, \tag{5}$$

the data term of p' is directly copied from its source pixel. Finally, update the source region Φ, the missing region Ω, and the filling front $\delta\Omega$. So far, a single iteration is finished. Repeat the above steps until Ω is completely filled.

Criminisi's algorithm has obtained relatively satisfactory results in filling large-area holes or removing objects in a picture, although shortcomings exit, some of them have been fixed by related works, they are listed as follows:

(1) The confidence term in the priority model may encounter a sharp decline after multiple iterations, while the fluctuation of the data term is relatively stable. Thus, the priority is more likely to be restricted by lower confidence term and become unreliable, leaving incorrect filling order and structural error propagation. Later, Zhou et al. [11] demonstrated that different weighted-priority should be chosen for specific structures to get better inpainting results. Liu [12] and Cao et al. [13] changed the priority formula into the exponential and addition form respectively to prevent the confidence term from falling too quickly. Xi et al. [14] eliminated the dependence on the shape of the target region and preserved the stability of confidence term by introducing the gray entropy;

(2) The similarity criterion between sample patch and target patch used in Criminisi's algorithm is the sum of squared differences (SSD) of corresponding pixels in two patches, which only takes the pixel value into account and does not make full use of the structural information. Martínez-Noriega et al. [15] added the Hellinger distance to measure the similarity of the probability distribution between two patches. Ran [16] introduced a metric of the structural similarity between two patches. Liu et al. [17] also defined a new match rule by taking structure tensor into consideration. These works have successfully reduced the rate of mismatch;

(3) Criminisi et al. chose a fixed patch size of 9×9 pixels, which is unreasonable, since the patch size directly affects its capability to capture the local texture features and has an important influence on inpainting quality. Different sizes of patch should be applied on regions with different uniformity levels. Generally, smaller patches should be applied on high-frequency areas with more textures

and structures to achieve a finer filling, while larger patches are appropriate for flatter areas to speed up the filling process, this idea was also demonstrated in Reference [1]. There are relatively few researches about this issue. Zhou et al. [18] determined the patch size automatically with gradient histograms involved, and this was implemented as an optimization problem that requires extra continuous iterations. However, this process is computationally expensive and takes a longer time;

(4) Global search in source region is required in Criminisi's method in order to find the best sample patch for the target patch, which needs a large amount of calculation and match time. Liu et al. [17] narrowed the match area by picking out those candidates whose sum of the pixel values is close to the target patch's, but this cannot guarantee that the bad candidates are excluded;

(5) Criminisi's and related patch-based inpainting techniques require a large number of iterations to completely fill the unknown area, since only one target patch can be filled in a single iteration. At present, there is no related research to handle this deficiency.

Compared with problems 1 and 2, there are relatively fewer studies on problems 3 and 4. Aiming at problems 3–5 mentioned above, we provide a novel solution: We first introduce the nonuniformity model, by which the patch size will be determined adaptively to address problem 3. Besides, the strategy of subregion search is proposed to address problem 4 in an effective way. Moreover, the strategy of a multi-patch match is proposed to handle problem 5 for the first time.

3. Proposed Approach

Our method uses a similar framework as Criminisi's [10]. Based on that, we add a step to evaluate the nonuniformity in an image at the beginning of the process. Based on nonuniformity and filling priority, multiple centers of the target patches are located on the filling front, patch sizes are determined by the nonuniformity, and each target patch's search area is limited from the global source region to particular subregion with similar content. After an iteration, these target patches will be filled with the content borrowed from the source region. Figure 2 briefly illustrates the basic framework of our work.

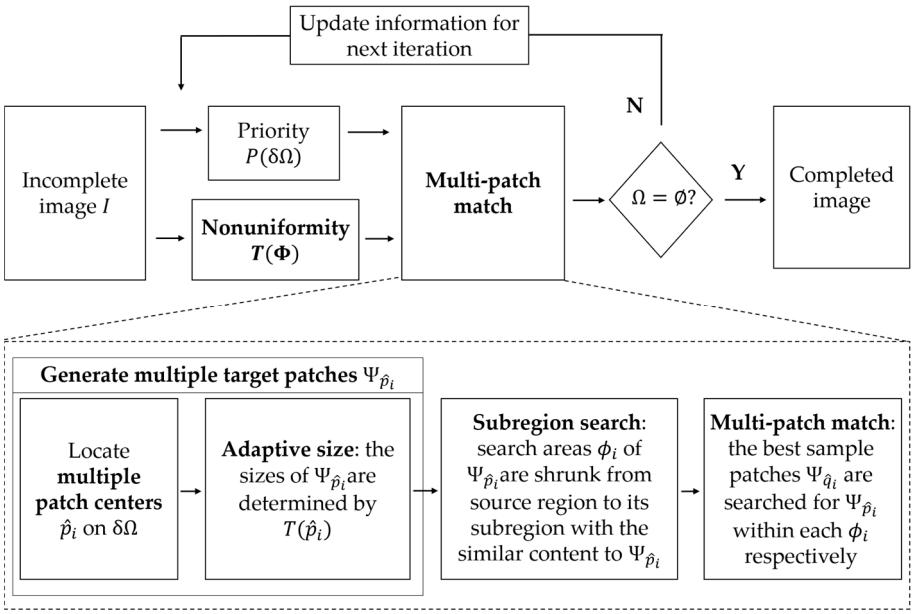

Figure 2. The overall framework of the proposed method. Our improvements are depicted in bold.

3.1. Evaluate the Nonuniformity

Evaluating the nonuniformity is the dependency of the other parts in our work, including the patch size determination, subregion search and multi-patch match. Therefore, the nonuniformity

needs to be quantified first. Given a patch of pixels that under a certain distribution, the standard deviation in statistics can effectively characterize the nonuniformity of pixel values, also can be seen as a measure of the texture feature [20]. Our nonuniformity is exactly based on the local standard deviation. Concretely, let S_p denote a square window centered at point p ($p \in \Phi$), w_s is the width of S_p determined by the image size $H \times W$, by default, $w_s = \max(2\left[\frac{\min(H,W)}{100}\right] + 1, 3)$, where $[\cdot]$ is the rounding operator. The local standard deviation at p is obtained by computing the standard deviation of all the known pixels in S_p:

$$\sigma(p) = \sqrt{\frac{\sum_{q \in S_p \cap \Phi} (q - \mu)^2}{|S_p \cap \Phi|}}, \tag{6}$$

where $\sigma(p)$ is the local standard deviation at p, μ is the mean value in S_p. Appling Equation (6) for all the pixels in source region, we obtain a map of local standard deviation $\sigma(\Phi)$, then $\sigma(\Phi)$ is normalized to the interval [0, 1] as the following equation:

$$\widetilde{\sigma}(\Phi) = \frac{\sigma(\Phi) - \min(\sigma(\Phi))}{\max(\sigma(\Phi)) - \min(\sigma(\Phi))}, \tag{7}$$

where $\widetilde{\sigma}(\Phi)$ is the normalized local standard deviation map. We find that in most cases if we directly use $\widetilde{\sigma}(\Phi)$ as the descriptor of the content without post-processing, its data distribution will be very uneven: The lower part is over-crowded and less distinguishable, while the higher part is too sparse (see Figure 3b for example). In Section 3.2, we intend to evenly divide the interval [0, 1] into subintervals so that every pixel can be categorized and be treated accordingly. Intuitively, it is better to stretch the crowded data to a relatively even distribution to fit the evenly-divided subintervals. We find that the widely-used histogram equalization is a simple and effective way to achieve this purpose, without any parameter to be set manually. Therefore, we equalize the $\widetilde{\sigma}(\Phi)$ to make full use of the space [0, 1]. Let histeq$(.)$ be the operation of histogram equalization:

$$T(\Phi) = \text{histeq}(\widetilde{\sigma}(\Phi)), \tag{8}$$

where T is the value equalized from $\widetilde{\sigma}$ and we define it as the nonuniformity. For instance, Figure 3c shows the nonuniformity map computed from Figure 3a.

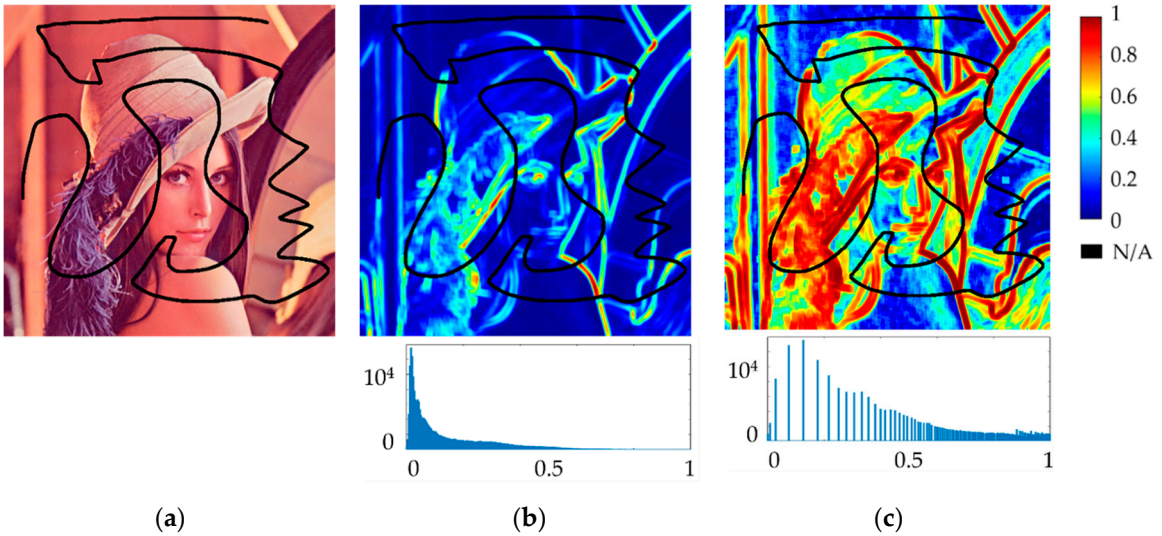

Figure 3. (**a**) Incomplete image Lena, the missing area is depicted in black; (**b**) map/histogram of $\widetilde{\sigma}(\Phi)$ of Lena (non-equalized); (**c**) map/histogram of equalized $\widetilde{\sigma}(\Phi)$ of Lena, defined as the nonuniformity $T(\Phi)$ in our context.

As shown in Figure 3c, each pixel p in Φ has a certain value of nonuniformity $T(p)$, which reflects the number of details contained in neighborhood of p. Intuitively, areas with more details (such as textured regions and line structures) will obtain a higher nonuniformity, whereas flatter areas (such as the background) will get a lower nonuniformity. For unknown areas, the nonuniformity makes no sense. If the unknown areas are filled with new pixels during the filling process, the nonuniformity of the new pixels is directly updated from their source pixels.

3.2. Adaptive Target Patch Size

Criminisi's algorithm utilizes a fixed patch size during the whole process. However, given an image, some areas contain rich textures and details, others may have little texture distribution. If a larger patch is used in rich-textured areas, stitching cracks are often easy to occur, thus losing the consistency of the texture structure. Therefore, the smaller patch should be used to achieve a smoother and natural texture propagation. Moreover, for those relatively flat regions, a larger patch is applied to prevent the staircase effect and speed up the filling process. It is more reasonable to apply different sizes of patch for different regions.

Denote n as the number of the sizes of patch used in our work. For a certain target patch $\Psi_{\hat{p}}$, its size $w_{\Psi_{\hat{p}}} \times w_{\Psi_{\hat{p}}}$ is specified by the nonuniformity $T(\hat{p})$, as the following equation:

$$w_{\Psi_{\hat{p}}} = -2\lfloor nT(\hat{p}) + 1 \rfloor + 2n + 3, \tag{9}$$

where $\lfloor \bullet \rfloor$ is downward rounding operator. The purpose of Equation (9) is to evenly divide the interval [0, 1] into n subintervals with the step of $\frac{1}{n}$. Suppose $T(\hat{p})$ falls in the k^{th} subinterval $\left[\frac{k-1}{n}, \frac{k}{n}\right), \ (k = 1, 2, \ldots, n)\Big)$, as k decreases from n to 1, $w_{\Psi_{\hat{p}}}$ increases from 3 to its maximum size $(2n + 1)$ with the step of 2 (the size should be odd), namely, the patch size is inversely proportional to the nonuniformity. Note that n should not be too small, or the choices of patch will be too limited to adaptively fit the different situations; neither too large, otherwise the max size of patch will also grow too large. If we paste an oversized patch into the target region, it is more likely to cause the stitching inconsistency even if in flat areas. We empirically let n change along with the image size $H \times W$ as $n = \max(\lfloor \frac{\min(H,W)}{100} \rfloor, 1)$, since we did not find the obvious evidence to show that there exists an optimal value of n.

3.3. Subregion Search

In Criminisi's algorithm, the global search for the best sample patch is required for every target patch, which is computationally expensive and unnecessary. In fact, it is more appropriate to let the target patch selectively match those sample patches that have similar content to the target patch, and skip those sample patches that are far different from the target patch. By doing so, firstly the search area will be narrowed from the entire source region to its subregion with the similar content to the target patch, so that the match time and the computation could be reduced. Moreover, a large number of unsatisfactory sample patches can be filtered out to reduce the mismatch, thereby improving the inpainting accuracy to a certain extent. Our "subregion search" strategy comes as follows.

Considering if the target patch center $\hat{p}$ is in a rich-textured area or strong edge, it is nearly impossible for those sample patches Ψ_q from poor-textured or flat area to serve as the ideal sample patches, instead, desired sample patches are supposed to have the similar content to $\Psi_{\hat{p}}$ and similar content means the similar nonuniformity level. Based on the above consideration, we narrow down the search area of $\Psi_{\hat{p}}$ from the entire source region to its subregion according to the nonuniformity map $T(\Phi)$ obtained by Equation (8). Specifically, the set of pixel q that satisfies the following equation is defined as the restricted subregion ϕ, which serves as the search area of the target patch $\Psi_{\hat{p}}$:

$$\phi = \Big\{ q \ \Big| \ \big|T(\hat{p}) - T(q)\big| \leq \alpha, \ q \in \Phi \Big\}, \tag{10}$$

where α is a parameter adjusts the strictness of limiting the search area. Smaller α means: (1) The subregion for search shrinks, the algorithm will be very careful with choosing the possible ideal candidates (see Figure 4a); (2) Lower fault-tolerance during the patch match (related to item 1), the method is more likely to miss the optimal candidate if search area is too small; (3) There will be more patch matches in an iteration because the space saved for each subregion makes room for more subregions (target patches) to get involved (see Section 3.4). By default, $\alpha = 0.1$. Figure 4b shows how subregion search works, the subregion ϕ for search is colored with respect to its target patch $\Psi_{\hat{p}}$ (blue patch, the size is magnified for clearer visualization). Only those sample patches whose center falls into subregion ϕ are considered potentially ideal (green patches), since they have the similar content to $\Psi_{\hat{p}}$, while those patches whose center outside the subregion ϕ are considered undesirable (red patches) and will be ignored during match process for they are far different from $\Psi_{\hat{p}}$. Intuitively, this strategy only allows the match between $\Psi_{\hat{p}}$ and a small portion of similar sample patches, which greatly saves the match time while maintaining the match accuracy.

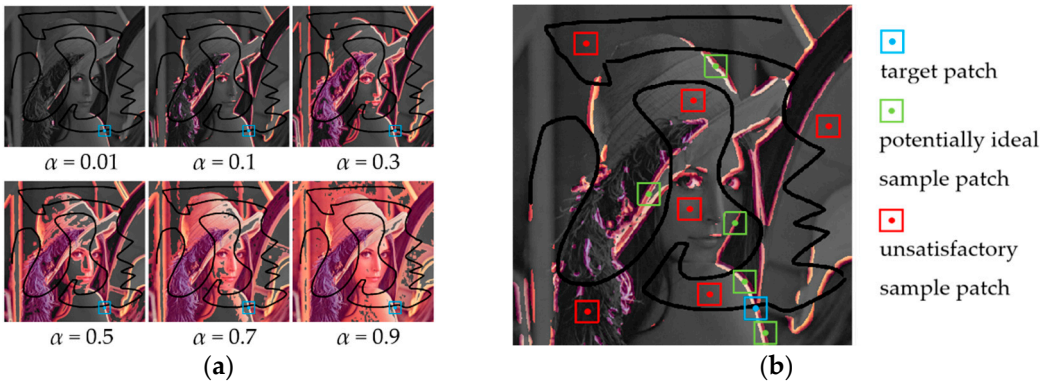

Figure 4. (**a**) Effect of parameter α on the subregion for search (colored area); (**b**) illustration for searching the ideal sample patches within the subregion ($\alpha = 0.1$).

3.4. Multi-Patch Match

During a single iteration of the existing patch-based methods, including Criminisi's, only one target patch can be filled after the match process, so that a large number of iterations are required to completely fill the missing area. To reduce the total number of iterations, we also propose the strategy of "multi-patch match": In each iteration, we appropriately select multiple pixels $\hat{p}_i$ $(i = 1, 2, 3, \ldots)$ with "the highest priority" on filling front, then generate multiple target patches $\Psi_{\hat{p}_i}$ centered at $\hat{p}_i$, and multiple best sample patches $\Psi_{\hat{q}_i}$ are also searched to fill $\Psi_{\hat{p}_i}$ correspondingly.

Criminisi et al. addressed that the filling order based on the priority model is crucial to prevent error inpainting. We also let the filling priority keep working in our approach. At first, the filling priority $P(\delta\Omega)$ on $\delta\Omega$ should be calculated by Equation (1). Based on this, we then select multiple pixels $\hat{p}_i$ with "the highest priority" on $\delta\Omega$ by considering its nonuniformity distribution $T(\delta\Omega)$. According to the idea that "target patch and its ideal sample patch should have similar content" mentioned in Section 3.3, and nonuniformity can be used as a scalar descriptor of content in a patch, match process of the two target patches with different nonuniformity levels can be considered independent, because the patch match in rich-textured regions or strong edges may not disturb the patch match in poor-textured or flat regions, and vice versa. Thanks to this, our multi-patch match is feasible. To reach this goal, a reasonable idea is to divide $\delta\Omega$ into multiple subsections with different nonuniformity intervals, let $\hat{p}_i$ be the highest priority pixel on each subsection, and multiple target patches $\Psi_{\hat{p}_i}$ are generated centered at $\hat{p}_i$. Specifically, we present the following algorithm shown in Algorithm 1 to show how to generate multiple target patches on filling front $\delta\Omega$.

Algorithm 1. Determination of Multiple Target Patches on Filling Front.

1. Initialization: let $i = 1$, find the filling front $\delta\Omega$, calculate its filling priority $P(\delta\Omega)$ using Equation (1).
2. Find the highest priority pixel $\hat{p}_i$ on $\delta\Omega$.
3. Define the subsection $\delta\Omega_i$ of $\delta\Omega$ as: $\delta\Omega_i = \left\{p \middle| \left|T(p) - T(\hat{p}_i)\right| \leq 2\alpha,\ p \in \delta\Omega\right\}$, then reset the priorities to zero for all pixels on $\delta\Omega_i$.
4. If there exits any non-zero priority pixels on $\delta\Omega$, let $i = i + 1$, back to step 2. Otherwise, go to step 5.
5. Generate target patches $\Psi_{\hat{p}_i}$ centered at $\hat{p}_i$ $(i = 1, 2, 3, \ldots)$, patch sizes are determined by Equation (9).

Note that we do not directly divide the filling front into subsections all at once, but in a progressive way that the current patch center $\hat{p}_i$ determines the current subsection $\delta\Omega_i$ and the next patch center $\hat{p}_{i+1}$ will be born outside the union of existing subsections $\delta\Omega_1 \cup \ldots \cup \delta\Omega_i$, namely, both subsections and patch centers are generated synchronously. This comes from the idea that the filling priority model always comes the first, then followed by subsection division. Illustration for above steps is shown in Figure 5. Figure 5f shows the final state, the nonuniformity of target patches $\Psi_{\hat{p}_i}$ are in different intervals, and each patch center also owns the highest filling priority on its subsection. As explained above, the order of $\Psi_{\hat{p}_i}$ may not be consistent with the descending order of their nonuniformity, but follows the order of filling priority.

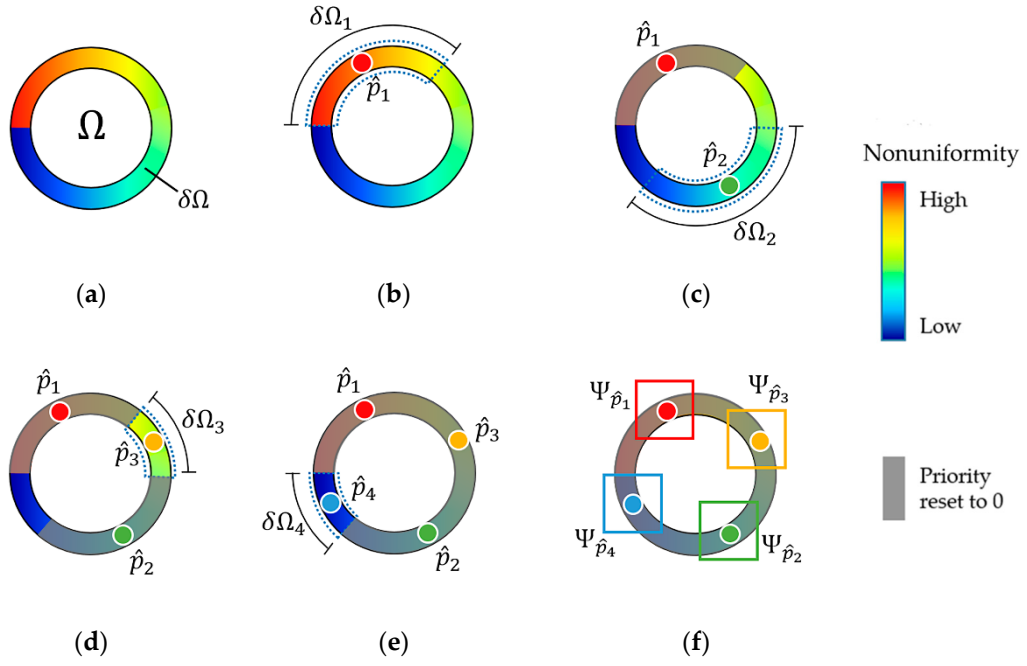

Figure 5. Illustration for determination of multiple target patches on filling front: (**a**) Filling front $\delta\Omega$; (**b**) find $\hat{p}_1$ with maximum priority, define the subsection $\delta\Omega_1$ on $\delta\Omega$ that has similar content to $\hat{p}_1$; (**c**) reset priorities on $\delta\Omega_1$ to 0, find the next pixel $\hat{p}_2$ with maximum priority, define $\delta\Omega_2$ that has similar content to $\hat{p}_2$; (**d**,**e**) continue to find all $\hat{p}_i$ and $\delta\Omega_i$ on $\delta\Omega$ until there is no non-zero priority pixel on $\delta\Omega$; (**f**) generate target patches $\Psi_{\hat{p}i}$ centered at $\hat{p}_i$.

After generating multiple target patches, the strategy of "subregion search" mentioned in Section 3.3 is also involved. For each target patch $\Psi_{\hat{p}_i}$, subregion ϕ_i with the similar content is assigned by Equation (10). As shown in Figure 6, during a single iteration, the patch match between Ψ_{q_i} and $\Psi_{\hat{p}_i}$ can proceed in their corresponding subregion ϕ_i , respectively, and each subregion ϕ_i will produce a best sample patch $\Psi_{\hat{q}_i}$ to fill its target patch. In this way, multiple patches can be filled in one iteration. By combining the subregion search and multi-patch match strategies, the speed of algorithm can be effectively improved.

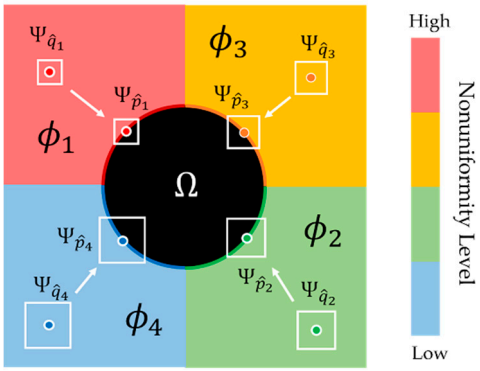

Figure 6. Illustration of a multi-patch match in single iteration.

It must be stressed out that step (2) in Algorithm 1 ensures that $\forall \hat{p}_i, \hat{p}_j \in \delta\Omega \ (i \neq j)$, the relation $\left|T(\hat{p}_i) - T(\hat{p}_j)\right| \geq 2\alpha$ always hold, in addition, Equation (10) guarantees that $\forall q \in \phi_i$, there always exists that $\left|T(q) - T(\hat{p}_i)\right| \leq \alpha$. These two equations will avoid the overlap of the nonuniformity interval and spatial scope between any two subregions, i.e., $T(\phi_i) \cap T(\phi_j) = \varnothing$, $\phi_i \cap \phi_j = \varnothing \ (i \neq j)$, ensuring that the match process in every subregion is independent of each other.

The description of our overall algorithmic steps is shown in Algorithm 2.

Algorithm 2. Overall Steps of Our Algorithm.

1. Initialization: compute the nonuniformity map $T(\Phi)$ as Equation (8).
2. Determine the filling front $\delta\Omega$, compute its priorities $P(\delta\Omega)$ as Equation (1).
3. Generate multiple target patches $\Psi_{\hat{p}_i}$ on $\delta\Omega$, as shown in Algorithm 1.
4. Assign subregion ϕ_i as the search scope for every $\Psi_{\hat{p}_i}$ as Equation (10).
5. Match the best sample patch $\Psi_{\hat{q}_i}$ for $\Psi_{\hat{p}_i}$ in ϕ_i using Equation (4).
6. Fill the unknown part of $\Psi_{\hat{p}_i}$ with corresponding pixels in $\Psi_{\hat{q}_i}$.
7. Update the data term for new pixels as Equation (5), confidence term and nonuniformity term are directly copied from their source pixels.
8. Update region Ω and Φ, if $\Omega = \varnothing$, exit the whole process. Otherwise, back to step 2.

4. Experimental Results

In this section, we evaluate the performance of the proposed method by conducting two types of experiments: Image restoration and object removal. The experiment is conducted on a computer with 2.2GHz CPU and 4 GB RAM, and implemented via MATLAB.

4.1. Image Restoration

4.1.1. Instance Test

We first show our inpainting results qualitatively and quantitatively by making a few instance tests. To make a comparison, several previous patch-based methods are also applied to our experiment, including Criminisi's exemplar-based inpainting method [10], Liu's method based on structure tensor [17], and Zhou's method using adaptive size based on gradient histograms [18]. One of the most representative diffusion-based methods, CDD model [4] proposed by Shen et al., is also involved in our test (3 k iterations employed). These instances include a portrait: Lena (512 × 512); two natural scenes: House (512 × 512), Sculpture (256 × 256); along with two pure texture images: Irregular pattern Texture I (640 × 640) and regular pattern Texture II (640 × 640) from Brodatz dataset [21,22]. These images are masked with black color that represents for missing areas. Both subjective and objective evaluation are compared among the proposed and previous methods mentioned above, where the subjective evaluation is the visual effects of completed images, as shown in Figures 7–11.

Figure 7. Comparison of inpainting results for Lena (512 × 512) (**a**) Ground truth; (**b**) incomplete image; (**c**) CDD model [4]; (**d**) Criminisi's method [10]; (**e**) Liu's method [17]; (**f**) Zhou's method [18]; (**g**) proposed method.

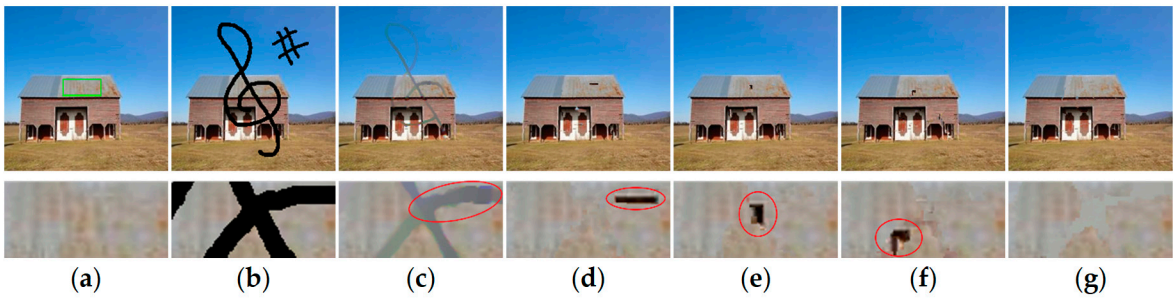

Figure 8. Comparison of inpainting results for House (512 × 512): (**a**) Ground truth; (**b**) incomplete image; (**c**) CDD model [4]; (**d**) Criminisi's method [10]; (**e**) Liu's method [17]; (**f**) Zhou's method [18]; (**g**) proposed method.

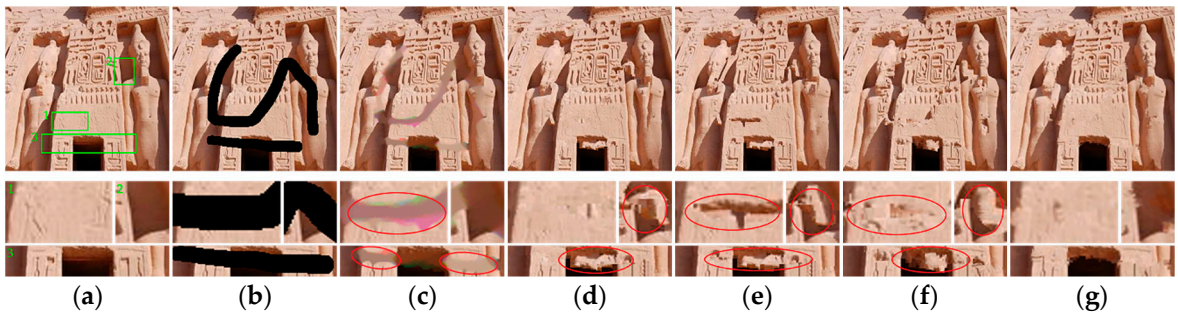

Figure 9. Comparison of inpainting results for Sculpture (256 × 256): (**a**) Ground truth; (**b**) incomplete image; (**c**) CDD model [4]; (**d**) Criminisi's method [10]; (**e**) Liu's method [17]; (**f**) Zhou's method [18]; (**g**) proposed method.

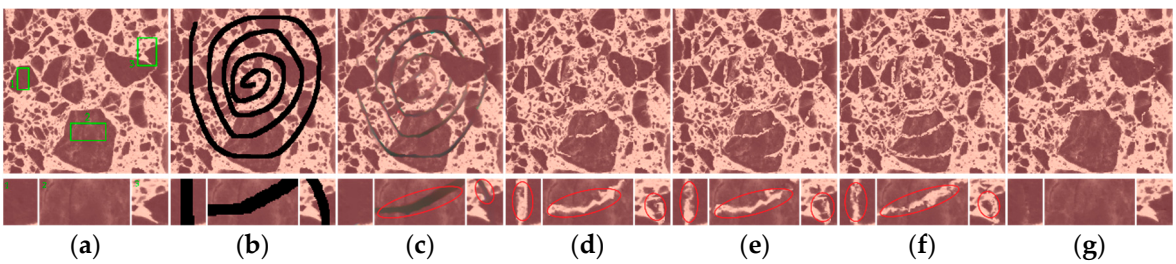

Figure 10. Comparison of inpainting results for irregular pattern Texture I (640 × 640): (**a**) Ground truth; (**b**) incomplete image; (**c**) CDD model [4]; (**d**) Criminisi's method [10]; (**e**) Liu's method [17]; (**f**) Zhou's method [18]; (**g**) proposed method.

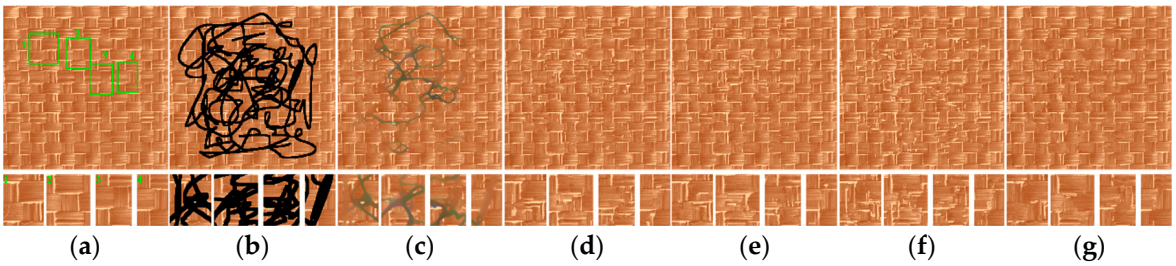

Figure 11. Comparison of inpainting results for regular pattern Texture II (640 × 640): (**a**) Ground truth; (**b**) incomplete image; (**c**) CDD model [4]; (**d**) Criminisi's method [10]; (**e**) Liu's method [17]; (**f**) Zhou's method [18]; (**g**) proposed method.

Subjective visual effects in Figures 7–11 show that compared with other methods, our results have obtained the least defects, and are most similar to the ground truth. As for the Lena (Figure 7), especially in those areas with rich textures and strong linear structures, such as the corner of eyes and the edge of the hat, our method will automatically generate a smaller target patch, ensuring that structures and textures are better preserved. In Reference [10], the fixed size of patch may not work well especially for areas with rich details, since the patch size can easily exceed the scale of the texture element, which could easily lead to stitching error and structure discontinuity. Reference [17] also uses a fixed size of patch, although a few of flaws still occur, benefiting from its improved priority model based on structure tensor, the algorithm has also achieved relatively good inpainting results. Reference [18] utilizes an adaptive size of patch based on gradient histogram; however, this method will generate a larger patch when connecting strong edges, as discussed above, this may lead to incorrect propagation of structures. Different from patch-based methods, Reference [4] does not copy patches from elsewhere, and performs better when dealing with such slim scratches by diffusion. As for the House (Figure 8), previous patch-based methods have occurred mismatch marked in red circles. This is because in the process of finding the best sample patch, both References [10] and [18] search the sample patch from the entire source region without filtering, and they are easier to match an inappropriate candidate if the match metric does not work well. In Reference [17], although the search area is limited by picking out those sample patches whose sum of pixel values falls within a certain range near its target patch's, however, this strategy does not always work well because the sum in a patch may reflect very limited information, those unsatisfactory sample patches may not be well excluded. Moreover, Reference [4] starts to show some diffusion artifacts as the scratches become thicker. In contrast, the proposed method searches for sample patch only in those areas that have similar content to the target patch, even if the similarity metric function loses its effect, it can still avoid selecting wrong sample patches to reduce mismatch. The Sculpture and Texture I, the rich-textured images with thick scratches in Figures 9 and 10, show that the structures and textures are well preserved in our results, whereas Reference [4] introduces noticeable blurring artifacts in such case and this is also known as the common issue for diffusion-based methods. Texture II (Figure 11) is a regular pattern, therefore, we believe additional steps are required to automatically perceive the scale of texture element for this periodically arranged textures to decide the optimal patch size before further improvements are made. Unfortunately, none of these methods has solved this challenging task yet.

The objective performance of algorithms will be assessed from two aspects: quality and efficiency. Peak signal to noise ratio (PSNR) and structural similarity index (SSIM) [23] are used to evaluate the similarity between the completed image and the ground truth, the higher PSNR and SSIM value means the higher quality of a completed image. Running time of the algorithm is used to measure inpainting efficiency. The objective evaluation for the above images under each algorithm is shown in Tables 1–3:

Table 1. Peak signal to noise ratio (PSNR) (dB) comparison of Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11.

Image	CDD [4]	Criminisi's [10]	Liu's [17]	Zhou's [18]	Proposed
Lena (512 × 512)	38.4142	35.0791	34.8686	35.5572	37.8445
House (512 × 512)	33.7388	34.4852	35.289	35.089	38.3214
Sculpture (256 × 256)	23.6974	24.0135	23.2412	23.3735	27.2959
Texture I (640 × 640)	22.4238	22.4142	22.4722	21.0718	24.1134
Texture II (640 × 640)	27.0897	28.4736	28.5973	26.2799	29.2464

Table 2. Structural similarity index (SSIM) comparison of Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11.

Image	CDD [4]	Criminisi's [10]	Liu's [17]	Zhou's [18]	Proposed
Lena (512 × 512)	0.9873	0.9775	0.9752	0.9752	0.9845
House (512 × 512)	0.9664	0.9807	0.9813	0.9823	0.9856
Sculpture (256 × 256)	0.8976	0.8928	0.8828	0.8808	0.9010
Texture I (640 × 640)	0.8686	0.8326	0.8323	0.7958	0.8780
Texture II (640 × 640)	0.8566	0.8930	0.8969	0.8347	0.9030

Table 3. Running time (s) comparison of Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11.

	CDD [4]	Criminisi's [10]	Liu's [17]	Zhou's [18]	Proposed
Lena (512 × 512)	113.66	306.49	238.91	574.85	81.02
House (512 × 512)	103.85	252.46	235.54	369.99	93.80
Sculpture (256 × 256)	44.38	24.02	22.93	39.70	9.56
Texture I (640 × 640)	371.84	955.24	721.95	1629.87	273.60
Texture II (640 × 640)	486.34	1373.26	1588.28	3204.63	421.28

From the perspective of objective evaluation, the inpainting quality reflected by PSNR and SSIM is basically consistent with the subjective visual perception. Restored areas with more coherent and natural textures and structures may present better visual effects, and will also obtain higher PSNR and SSIM values. As for inpainting efficiency, our method has successfully reduced the time consumption thanks to our subregion search and multi-patch match strategies. Although our method is able to fill a maximum of six target patches in a single iteration, the running time is not shortened by six times as expected. This is because when the target patch size adaptively gets smaller, the inpainting progress will be slowed down. We also notice that if the image size gets larger, the acceleration becomes more obvious, since the average patch size will also increase. While Reference [17] tries to limit the search area by picking out those candidates whose sum is close to the target patch's, this does not significantly reduce the calculation and obtains limited acceleration. References [10] and [18] both adopt global search; and Reference [18] converts the determination of optimal patch size into an extra optimization problem, that requires additional iterations and decreases the overall efficiency.

4.1.2. Batch Test

In order to evaluate the performance of these algorithms on a broader level, our next experiment is conducted with more test samples. We randomly select 100 images from the public dataset Places2 [24] and resize them to 512 × 512. Then we make 20 masks to randomly generate missing areas on those sample images. The aforementioned methods are tested, and their objective quantitative evaluation indicators: PSNR, SSIM, and running time are plotted in Figures 12a, 13a and 14a, respectively.

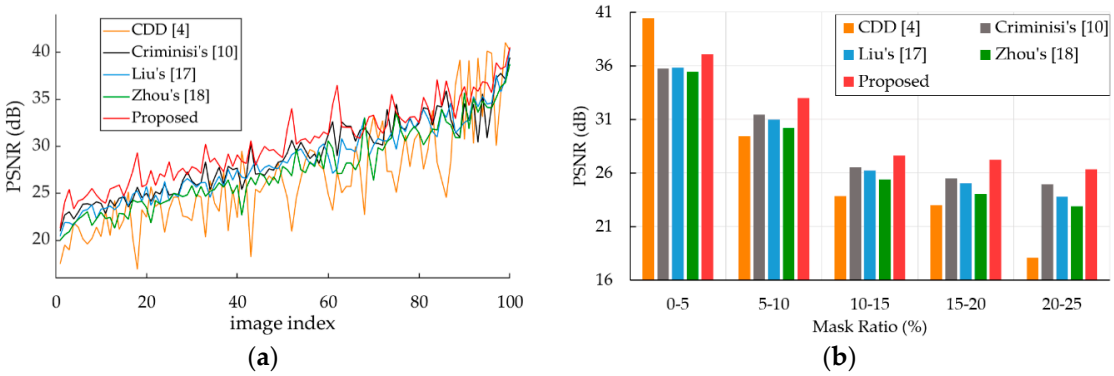

Figure 12. Comparison of PSNR for different methods: (**a**) PSNR of 100 inpainted results; (**b**) effect of mask ratios on the mean PSNR.

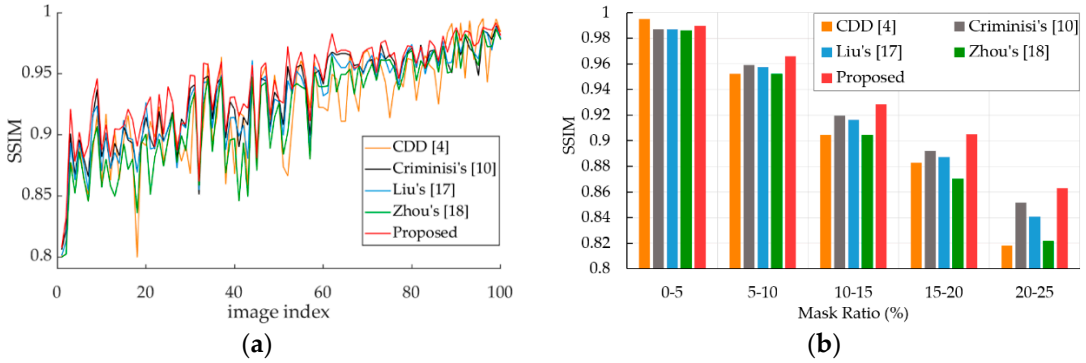

Figure 13. Comparison of SSIM for different methods: (**a**) SSIM of 100 inpainted results; (**b**) effect of mask ratios on the mean SSIM.

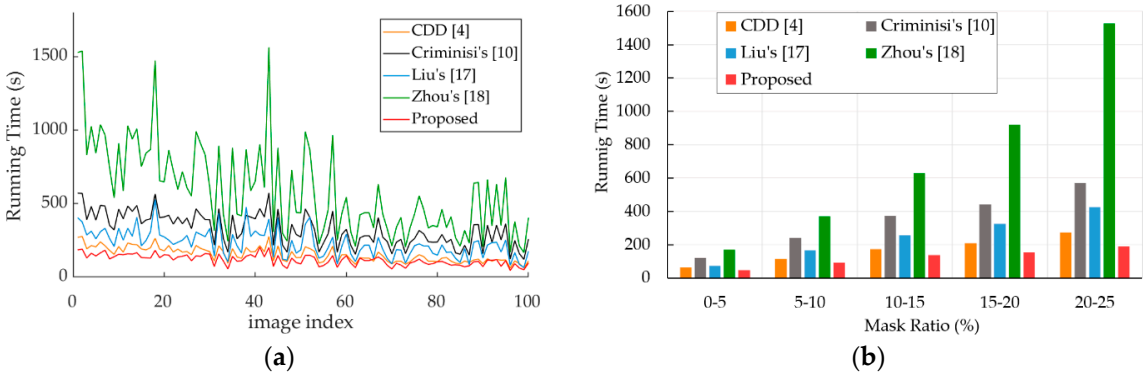

Figure 14. Comparison of running time for different methods. (**a**) Running time of 100 inpainted results; (**b**) Effect of mask ratios on the mean running time.

We also study the performance of these methods in relation to the percentage of the masked area from above 100 samples. These samples are divided into five categories with respect to their mask ratios. Mask ratios are divided into five intervals vary from 0% to 25% with the step of 5%. Figures 12b, 13b and 14b show how the mask ratios affect the mean PSNR, SSIM, and running time of each method.

It can be learned from Figures 12a and 13a that PSNR and SSIM curves are very close that are not so distinguishable. The reason is that the differences between the restored images are those masked areas only, which are much smaller than the source region that is exactly the same as the original image. However, it still can be seen that, for most images, the proposed method has obtained relatively higher PSNR and SSIM values than other methods. In fact, our method achieves the highest PSNR

value in 78 samples out of 100, the highest SSIM value in 67 samples out of 100, and the shortest running time in 97 samples out of 100. Figures 12b and 13b suggest that the CDD model could obtain relatively better performance when handling smaller mask size, but once as the mask size increases, this diffusion-based method may struggle at restoring the expected content and begin to fall behind the patch-based methods, whereas the proposed method achieves the best average performance in most cases. In order to make a clearer comparison of the overall performance of five algorithms from a quantitative perspective, the overall average values of those curves in Figures 12a, 13a and 14a are recorded in Table 4.

Table 4. Average values in Figures 12–14.

Method	Mean PSNR (dB)	Mean SSIM	Mean Running Time (s)
CDD [4]	26.5482	0.9218	152.81
Criminisi's [10]	28.9099	0.9321	324.79
Liu's [17]	28.4467	0.9290	229.21
Zhou's [18]	27.5921	0.9193	591.42
Proposed	30.3950	0.9408	117.49

Compared with Criminisi's algorithm [10], the average PSNR and SSIM in our algorithm are improved by 5.14% and 0.93% respectively, and the efficiency is improved by 276%. Our results also outperform the results in References [4,17,18].

In summary, the proposed approach has made effective improvements in dealing with the deficiencies of the previous patch-based inpainting algorithm, both subjectively and objectively.

4.2. Object Removal

Object removal is another typical application of image inpainting, by replacing the unwanted object with a plausible background in an image. Figure 15 shows examples where we attempt to remove unwanted objects from the existing images by using our method. Different from damaged image restoration, this task does not have ground truth as the reference, and there also might be multiple possible filling solutions. Thus, the objective quality evaluation is no longer applicable, here, we only provide the visual results. Note that as the masked area increases, some artifacts may still occur, such as the unnatural water texture in Figure 15f, and the discontinuity of the curved edge of the lawn in Figure 15g.

(**a**) (**b**) (**c**) (**d**) (**e**) (**f**) (**g**)

Figure 15. Examples of object removal using our method. Top row—original images; middle row—masks for the unwanted part; bottom row—results for object removal.

5. Conclusions and Future Work

In this paper, a novel multi-patch-based image inpainting algorithm is proposed for filling missing areas in a more accurate and efficient way. Aiming at the shortcomings that traditional patch-based methods use a fixed size of patch and a global search for finding the best sample patch, and only one target patch can be filled in a single iteration, which is computationally expensive, we provide a novel solution: We first introduce a measurement model to quantify the nonuniformity in an image, then different sizes of patches are adaptively determined for regions with different nonuniformity levels, making the restored textures and structures more coherent and natural so that inpainting quality is improved. Moreover, by fully utilizing the nonuniformity, the source region is divided into multiple non-overlapping subregions with different nonuniformity levels; and in each subregion, the best sample patch is matched for target patch. This has successfully reduced the match time in a single iteration and the total number of iterations, as well as the rate of mismatch. Experimental results show that our improved algorithm has obtained better inpainting quality, both subjectively and objectively with less time-consuming.

In addition, in terms of the inpainting quality, if related works are combined, such as improved priority model and match criterion, the results may become better. In terms of inpainting efficiency, further improvements can be achieved based on the acceleration strategy mentioned in this article. For example, the scalar value of nonuniformity used in this article may contain limited information, other features, such as texture directions, colors, etc., can also be introduced in the process of narrowing the search area to achieve further acceleration.

Finally, there are also limitations in our algorithm and other patch-based inpainting algorithms. These methods assume that the texture in the missing region can be found elsewhere in the source region. However, this assumption does not always hold—once the missing information is locally unique, similar structures cannot be found, these methods may struggle at reconstructing satisfactory results. Fortunately, in recent years, deep learning techniques have been introduced, and hopefully they are capable of making up for this deficiency. For example, Nazeri et al. [25] utilized two-stage GAN that has achieved impressive results. Image inpainting based on deep learning techniques might be a novel and robust way—especially in those complex cases and are worth further study.

Author Contributions: Conceptualization, S.Y. and H.L.; Data curation, S.Y.; Formal analysis, H.C.; Investigation, S.Y. and H.L.; Methodology, S.Y.; Project administration, Y.W. and X.C.; Software, S.Y.; Supervision, H.C. and X.C.; Validation, S.Y. and H.L.; Writing—original draft, S.Y., H.L., Y.W. and X.C.; Writing—review & editing, S.Y., H.C. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the Tianjin Municipal Transportation Commission Science and Technology Development Plan Project (2019C-05).

Acknowledgments: The authors would like to thank Zhou Bolei and Abdelmounaime Safia et al. for the open-access dataset they provided.

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

References

1. Chen, X.D.; Liang, H.T.; Xu, H.Y.; Ren, S.Y.; Cai, H.Y.; Wang, Y. Artifact handling based on depth image for view synthesis. *Appl. Sci.* **2018**, *9*, 1834. [CrossRef]
2. Bertalmio, M.; Sapiro, G.; Caselles, V.; Ballester, C. Image inpainting. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 23–28 July 2000; pp. 417–424.
3. Chan, T.F.; Shen, J.H. Mathematical models for local non-texture inpainting. *SIAM J. Appl. Math.* **2002**, *62*, 1019–1043. [CrossRef]
4. Shen, J.H.; Kang, S.H.; Chan, T.F. Euler's Elastica and curvature-based inpainting. *SIAM J. Appl. Math.* **2003**, *63*, 564–592. [CrossRef]

5. Oliveira, M.; Bowen, B.; McKenna, R.; Chang, Y.S. Fast Digital Image Inpainting. In Proceedings of the Visualization, Imaging, and Image Processing IASTED Conference, Marbella, Spain, 3–5 September 2001; pp. 261–266.
6. Garber, D.D. Computational models for texture analysis and texture synthesis. *Proc. SPIE* **1981**, *281*, 254–273. [CrossRef]
7. Ashikhmin, M. Synthesizing natural textures. In Proceedings of the 2001 Symposium on Interactive 3D Graphics, New York, NY, USA, 19–21 March 2001; pp. 217–226.
8. Liang, L.; Liu, C.; Xu, Y.Q.; Guo, B.N. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.* **2001**, *20*, 127–150. [CrossRef]
9. Efros, A.A.; Freeman, W.T. Image quilting for texture synthesis and transfer. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 12–17 August 2001; pp. 341–346.
10. Criminisi, A.; Perez, P.; Toyama, K. Region filling and object removal by exemplar-based inpainting. *IEEE Comput. Trans. Image Process.* **2004**, *13*, 1200–1212. [CrossRef] [PubMed]
11. Zhou, Y.; Li, L.; Xia, K. Research on weighted priority of exemplar-based image inpainting. *J. Electron. (China)* **2012**, *29*, 166–170. [CrossRef]
12. Liu, Y.F.; Wang, F.L.; Xi, X.Y.; Liu, Z.H. Improved algorithm for image inpainting based on texture synthesis. *J. Chin. Mini-Micro Comput. Syst.* **2014**, *35*, 2754–2758. [CrossRef]
13. Cao, Q.; Tang, W.; Wan, T.R.; Zhu, Y.L.; Wu, T. Image restoration based on Criminisi algorithm for improving priority. *Video Eng.* **2019**, *43*, 110–115.
14. Xi, X.Y.; Wang, F.L.; Liu, Y.F. Improved Criminisi algorithm based on a new priority function with the gray entropy. In Proceedings of the International Conference on Computational Intelligence & Security, Le Shan, China, 19–21 December 2013; pp. 214–218.
15. Martínez-Noriega, R.; Roumy, A.; Blanchard, G. Exemplar-based image inpainting: Fast priority and coherent nearest neighbor search. In Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, Santander, Spain, 23–26 September 2012; pp. 1–6.
16. Ran, D.P. An Improved Criminisi Algorithm for Image Inpainting. Master's Thesis, Hunan University, Changsha, China, May 2015.
17. Liu, Y.; Liu, C.J.; Zou, H.L.; Zhou, S.S.; Shen, Q.; Chen, T.T. A novel exemplar-based image inpainting algorithm. In Proceedings of the 2015 International Conference on Intelligent Networking and Collaborative Systems (INCOS), Taipei, Taiwan, 2–4 September 2015; pp. 86–90.
18. Zhou, H.L.; Zheng, J. Adaptive patch size determination for patch-based image completion. In Proceedings of the IEEE 2010 17th IEEE International Conference on Image Processing, Hong Kong, China, 9 September 2010; pp. 421–424.
19. Barnes, C.; Shechtman, E.; Finkelstein, A.; Goldman, D.B. Patch match: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* **2009**, *28*, 24. [CrossRef]
20. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 3rd ed.; Electronic Industry Press: Beijing, China, 2017; pp. 434–436.
21. Brodatz, P. *Textures: A Photographic album for Artists and Designers*; Dover Publications: New York, NY, USA, 1999.
22. Safia, A.; He, D. New Brodatz-based Image Databases for Grayscale Color and Multiband Texture Analysis. *ISRN Mach. Vis.* **2013**, *2013*. [CrossRef]
23. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
24. Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; Torralba, A. Places: A 10 million image database for scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 1452–1464. [CrossRef] [PubMed]
25. Nazeri, K.; Ng, E.; Joseph, T.; Qureshi, F.; Ebrahimi, M. EdgeConnect: Generative image inpainting with adversarial edge learning. *arXiv* **2019**, arXiv:1901.00212.