IEICE TRANS. INF. & SYST., VOL.E96–D, NO.12 DECEMBER 2013

LETTER

Pixel and Patch Reordering for Fast Patch Selection in Exemplar-Based Image Inpainting

Baeksop KIM[†], **Jiseong KIM**[†], ***Nonmembers***, ***and*** **Jungmin SO**[†a)], ***Member***

SUMMARY This letter presents a scheme to improve the running time of exemplar-based image inpainting, first proposed by Criminisi et al. In the exemplar-based image inpainting, a patch that contains unknown pixels is compared to all the patches in the known region in order to find the best match. This is very time-consuming and hinders the practicality of Criminisi's method to be used in real time. We show that a simple bounding algorithm can significantly reduce number of distance calculations, and thus the running time. Performance of the bounding algorithm is affected by the order of patches that are compared, as well as the order of pixels in a patch. We present pixel and patch ordering schemes that improve the performance of bounding algorithms. Experiments with well-known images used in inpainting literature show that the proposed reordering scheme can reduce running time of the bounding algorithm up to 50%.

key words: *image inpainting, image completion, exemplar-based, bounding, fast nearest-neighbor algorithm*

1. Introduction

Image inpainting is a technique to fill in unknown regions in an image. The applications of image inpainting include removing scratches and stains, recovering missing parts of the image, and removing an unwanted object from the image and replacing with the background plausible to the human eye. This problem has gained significant interest since Bertalmio et al. [2], and many different solutions have been proposed. The solutions to image inpainting can be largely categorized into two classes: PDE (Partial Differential Equation)-based and exemplar-based. The PDE-based methods [2], [3], [6], basically fill the missing region by diffusing known pixels in the neighborhood while trying to maintain structures. This is done by solving third or fourth-order partial differential equations iteratively. The PDE-based methods are good at filling small gaps such as scratches but do not provide good results if the size of the missing region is big. Criminisi et al. [1] proposed the exemplar-based inpainting method, where the unknown region is iteratively filled by selecting the "best" patch from the known region. The filling order is selected according to a priority function, which favors maintaining the isophote direction in the known region. The exemplar-based method is well suited for filling in large gaps, such as removing unwanted objects from the image. Many improvements to the original solution have been proposed in the line of exemplar-based approach, in order to enhance the resulting image [4], [5], [8].

The exemplar-based inpainting method [1] works as follows. Define Φ as the source region (known region) and Ω as the target region to be filled. The contour of the target region is called "filling front", defined by $\delta\Omega$. Now we pick an image patch Ψ_p centered at pixel p, where p is a pixel in the filling front. Ψ_p is called the *target patch*. We search for an image patch in the known region that is the most similar to Ψ_p. The best-matched image patch, Ψ_q replaces Ψ_p, and this finishes a single iteration. The iterations continue until all pixels in the missing region are filled. The order of selecting Ψ_p is of critical importance, since it will affect how well the structures will be maintained. The priority of the patch is defined as $P(p) = C(p)D(p)$, where $C(p)$ is called the confidence term and $D(p)$ is called the data term. $C(p)$ and $D(p)$ are defined as follows.

$$C(p) = \frac{\sum_{q\in\Psi_p\cap\Omega} C(q)}{|\Psi_p|}, \qquad D(p) = \frac{\nabla I^{\perp}_{n_p} \cdot n_p}{\alpha} \tag{1}$$

In the above equation, $|\Psi_p|$ is the area of Ψ_p, α is the normalization factor. n_p is the unit normal vector orthogonal to $\delta\Omega$ at p, and $\nabla I^{\perp}_{n_p}$ is the isophote at p. The confidence term $C(p)$ is updated after every iteration.

In each iteration, the to-be-filled image patch Ψ_p must be compared to every patch centered at $q \in \Phi$ (called *candidate patch*). To compare similarity of the two patches, SSD (Sum of Squared Differences) of the known pixels is used. This is a very time-consuming job, and it hinders the practicality of the method to be used in real-time on small devices such as smartphones. Efforts have been made to speed up the process of finding the best patch. The most simple method is to restrict search area to the neighborhood of the unknown patch, assuming that the best patch will be mostly found near the unknown patch [7]. Another method [9] decomposes image into frequency coefficients and selects only significant coefficients, thereby speeding up the matching process. The two methods both try to improve time complexity by neglecting insignificant information. Although they can speed up the matching process, the result can be different from Criminisi's result. Our focus here is to speed up the process while maintaining the same result as Criminisi's. We use a bounding algorithm to remove unnecessary comparisons. In addition, we order patches and pixels to be compared such that the speed of the bounding algorithm is improved. We discuss the bounding algorithm, pixel reordering and patch reordering schemes in the following sections, along with their effects on time com-

Manuscript received June 3, 2013.
Manuscript revised August 7, 2013.

[†]The authors are with the Dept. of Computer Engineering, Hallym University, Korea.
a) E-mail: jso1@hallym.ac.kr (Corresponding author)

DOI: 10.1587/transinf.E96.D.2892

plexity of the exemplar-based inpainting.

2. Bounding Algorithm for Fast Search

Suppose the target patch Ψ_p has been selected, and we need to find the most similar patch in the known region using the SSD metric. The patch size is an algorithm parameter, and we use 9×9 pixels as in [1]. For each known pixel c_i in Ψ_p, the color difference δ_i is calculated against the pixel c_i' in the candidate patch Ψ_q as follows. In the equation, $c_{i,R}$, $c_{i,G}$, $c_{i,B}$ are R, G, B values of pixel c_i, respectively.

$$\delta_i = (c_{i,R} - c'_{i,R})^2 + (c_{i,G} - c'_{i,G})^2 + (c_{i,B} - c'_{i,B})^2 \quad (2)$$

We call δ_i the *pixel distance*. Now, the *patch distance* between Ψ_p and Ψ_q denoted as $D_{p,q}$, is as follows.

$$D_{p,q} = \sum_{i,c_i \in \Psi_p \cap (I-\Omega)} \delta_i \quad (3)$$

For example, if the number of known pixels in Ψ_p is k, we need to calculate δ value k times for each candidate patch in the known region. However, if we know the patch distance with the current best patch, we can reduce the number of calculating pixel distances. Suppose the current best patch is q_B. When we compare Ψ_p and Ψ_q, we can reduce calculations by bounding the patch distance at D_{p,q_B}. The patch comparison process is shown as a pseudocode in Fig. 1.

This simple bounding algorithm can reduce significant amount of calculation, as shown later in Table 3. In this bounding algorithm, the candidate patch was selected in the sequential order, starting from the patch in the top-left corner. However, the performance of bounding algorithm depends on the order of comparing pixels and patches. In the next sections, we propose pixel and patch reordering techniques to improve performance of the bounding and algorithm and further reduce computation time of inpainting.

3. Pixel Reordering Scheme

When comparing two patches, the pixel distance of corresponding pixels in the target and candidate patch are summed up. If we have an upper bound, which is the patch distance between the target patch and the current best patch, then we can stop adding up once the sum exceeds the upper bound. Since the upper bound is fixed while computing patch distance, the best strategy is to calculate pixel distances in the descending order of pixel distance. In other words, we want to order the pixel distance calculations so that we can quickly reach the upper bound. However, we do not know beforehand which pixel in the target patch will show a large distance from the corresponding pixel in the candidate patch. Thus, we use a statistical method to order the pixels.

First we create a histogram by counting the number of pixels for each color value in the image. Then, when comparing two patches, we start from the pixel with color that has the least frequency. Here we use the gray-scale value (0 to 255) when counting the number of pixels for each color. There are alternative ways to choose the pixels; (1) count the number of pixels for each (R, G, B) value, (2) count the number of pixels separately for RGB, add the counts, and choose the pixel with the minimum sum, (3) count the number of pixels separately for RGB and choose the pixel with the minimum count regardless of RGB. The first alternative requires too much memory to store the histogram. The second and third alternatives achieve similar performances with the gray-scale approach, although the exact numbers were omitted due to lack of space.

For each patch Ψ_q in $(I - \Omega)$
  $D_{p,q} \leftarrow 0$
  For each pixel c_i in $\Psi_p \cap (I - \Omega)$
    $\delta_i = (c_{i,R} - c'_{i,R})^2 + (c_{i,G} - c'_{i,G})^2 + (c_{i,B} - c'_{i,B})^2$
    $D_{p,q} \leftarrow D_{p,q} + \delta_i$
    If $D_{p,q} > D_{p,q_B}$, break
  If $D_{p,q} < D_{p,q_B}$
    $q_B \leftarrow q$
    $D_{p,q_B} \leftarrow D_{p,q}$

Fig. 1 A bounding algorithm to find the best patch.

Table 1 The test images used in performance evaluation of the proposed schemes.

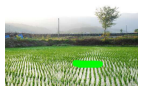
Title	Original	Marked	Inpainted
Rein River			
Rice Field			
Woman			
Pumpkin			
Golf			
Bungee			
Cable Car			

Table 2 Correlation between frequency of gray-scale color and average pixel distance.

Title	Correlation
Rein River	−0.1714
Rice Field	−0.7235
Woman	−0.6794
Pumpkin	−0.4708
Golf	−0.2145
Bungee	−0.3219
Cable Car	−0.7321

$S = \Psi_p \cap (I - \Omega)$
while S is not empty
  Find pixel c_m in S where $c_m = \arg\min H(c_m)$
  Remove c_m from S
  $\delta_m = (c_{m,R} - c'_{m,R})^2 + (c_{m,G} - c'_{m,G})^2 + (c_{m,B} - c'_{m,B})^2$
  $D_{p,q} \leftarrow D_{p,q} + \delta_m$
  If $D_{p,q} > D_{p,q_B}$, stop

Fig. 2 Calculating patch distance using pixel reordering.

The rationale behind using the histogram is that the pixel distance will be large with high probability when comparing pixels with infrequently appearing color. To find out if this is indeed true, we compute the correlation between average pixel distance and the frequency of gray-scale color that appears in the image. We use seven test images, which include the images used in [1]. The images and their inpainted results are shown in Table 1, and the correlation values are shown in Table 2. The **Rein River** image has the highest correlation value at −0.1714, and other images have lower correlation value. This means that the frequency of color has negative relationship with pixel distance. In other words, if the color of a pixel appears less frequently in the image, its pixel distance with another pixel tends to be higher. Thus, selecting pixels in the increasing order of color frequency can be beneficial.

Figure 2 shows the pseudocode for computing patch distance using the pixel reordering. In Fig. 2, $H(c_i)$ is the number of pixels in $(I - \Omega)$ which has the same gray-scale color value as pixel c_i. Note that we only need to calculate the frequency of each gray-scale color once for inpainting the whole image.

4. Patch Reordering Scheme

The pixel reordering scheme reduces number of calculations by making the sum quickly reach the upper bound. If the sum of pixel distances does not reach the upper bound after processing all known pixels, it means that a new upper bound is found, thus the upper bound is updated. Another way to speed up computation is to let the upper bound quickly decrease, so that patch distances are computed with the lowest upper bound as possible. This is done by reordering the patch distance calculation. As discussed earlier, some previous works have restricted search area to the neighborhood of the target region, because the best matching patch is likely to be near the target patch [7]. This argument is intuitive, but cannot be proved to be true for all images. Thus, we do not restrict the search area. Instead, we order the patches so that the ones near the target patch is processed earlier than the ones that are far away.

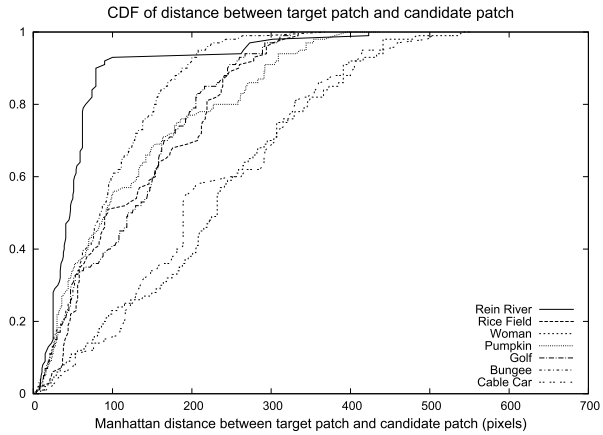


Fig. 3 CDF of Manhattan distance between target patch and its best-matching patch.

To see if this strategy is reasonable, we plot the cumulative distribution function (CDF) of Manhattan distance between center of the target patch and center of the best candidate patch. If the distance is short for most cases, searching from the nearer patches will be a good strategy. The CDF is shown in Fig. 3. For five out of seven images, the best patch was located less than 150 pixels away from the target patch in 70% of the target patches. Considering the image sizes (shown in Table 3), we can say that the best patches are geometrically close to the target patch in typical cases. However, this is not true for **Woman** and **Cable Car**. For more than 50% of the target patches, the best patch is located more than 200 pixels away. As shown in Fig. 5, patch reordering does not give much improvement for these two images. Note that in cases such as **Woman** and **Cable Car**, restricting search area to the region near the target patch as in [7] may degrade the resulting image.

Figure 4 shows the simple pseudocode for patch reordering. Basically, the candidate patches are ordered so that the one closest to the target patch is processed first. In the pseudocode, $M(p, q)$ is the Manhattan distance between pixel p and q and is defined as $M(p, q) = |p_x - q_x| + |p_y - q_y|$, where p_x and p_y are x and y coordinates of pixel p, respectively.

5. Performance Evaluation

We have evaluated performance of the proposed schemes using seven images shown in Table 1. Table 3 shows the result in numbers and Fig. 5 shows the computation time normalized to the performance of bounding algorithm without pixel or patch reordering. First, we can see that just applying bounding algorithm significantly reduces the number of pixel distance calculations. In addition, applying pixel and patch reordering can further reduce computation time by 15-

Table 3 Number of pixel color difference calculations for various schemes. Units are in millions.

Title	Size	Criminisi's	Bounding algorithm	Bounding with pixel reordering	Bounding with pixel & patch reordering
Rein River	500 × 375	884.60	98.67	90.12	62.84
Rice Field	500 × 334	892.34	175.29	116.30	105.46
Woman	360 × 480	9399.88	1102.92	900.18	942.39
Pumpkin	472 × 332	2275.96	295.40	268.15	176.90
Golf	350 × 262	1287.17	187.07	151.03	133.18
Bungee	206 × 308	877.10	191.67	108.63	98.43
Cable Car	342 × 450	1188.42	276.23	174.02	172.06

For each patch Ψ_p
  $S = \{q | q \in (I - \{p\})\}$
  While S is not empty
    Select q in S where $q = \arg\min_q M(p, q)$
    Remove q from S
    If $\Psi_q \subset (I - \Omega)$
      Compute $D_{p,q}$

Fig. 4 The patch reordering algorithm. Patches closer to the target patch are processed first.

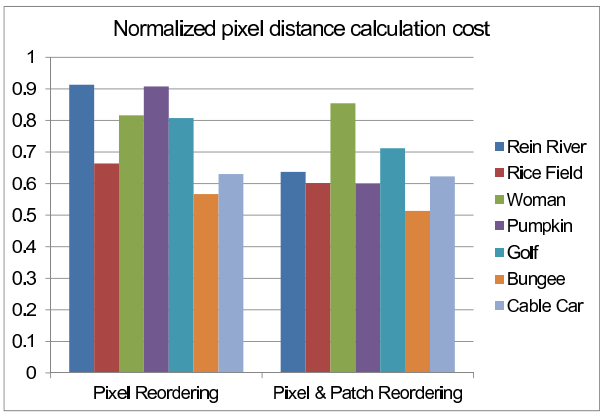


Fig. 5 Computation cost for bounding with pixel reordering and bounding with pixel & patch reordering, normalized to the cost of bounding algorithm with no reordering.

50%. For some images such as **Bungee** and **Cable Car**, the pixel reordering greatly helps speeding up the search. For other images such as **Rein River** and **Pumpkin**, patch reordering plays a crucial role in improving performance. The two schemes can be synergically combined to speed up the process of searching for the best patch. However, in some cases such as **Woman**, patch reordering increases the computation time. The reason can be deducted from Fig. 3, which shows that the best patch in **Woman** is not geometrically close to the target patch. In another example of this kind, **Cable Car**, applying patch reordering does not improve computation time significantly. Nevertheless, even in these two cases, patch reordering does not significantly worsen the performance. In summary, pixel reordering and patch reordering can reduce running time of inpainting in most cases. In other cases they can achieve similar performance with bounding with no reordering. The amount of performance improvement depends on the characteristics of the inpainted image.

6. Conclusion

Image inpainting is a useful technique in image processing, from removing scratches and stains to removing unwanted objects from the image. An examplar-based inpainting algorithm is quite successful in achieving the goal, but its long processing time hinders its practicality in real time. In the exemplar-based algorithm, the most time-consuming process is to search for the best image patch that will replace the target patch in the fill front. We propose schemes that remove unnecessary calculations as much as possible, while maintaining the same result as the original algorithm. By ordering distance calculations of pixels and patches in a certain way, we can improve the performance of bounding algorithm significantly. Experiments with well-known images in inpainting literature show that the pixel and patch reordering schemes combined can speed up the search process up to 50% compared with bounding with no reordering.

References

[1] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," IEEE Trans. Image Process., vol.13, pp.1200–1212, Sept. 2004.
[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," ACM SIGGRAPH, pp.417–424, 2000.
[3] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," ACM Trans. Graphics, vol.22, no.3, pp.303–312, 2003.
[4] N. Komodakis and G. Tziritas, "Image completion using global optimization," IEEE CVPR, pp.442–452, 2006.
[5] J. Sun, L. Yuan, J. Jia, and H. Shum, "Image completion with structure propagation," ACM Trans. Graphics, vol.24, no.3, pp.861–868, 2005.
[6] T. Chan, S. Osher, and J. Shen, "The digital tv filter and nonlinear denoising," IEEE Trans. Image Process., vol.10, no.2, pp.231–241, 2001.
[7] Anupam, P. Goyal, and S. Diwakar, "Fast and Enhanced Algorithm for Exemplar Based Image Inpainting," Proc. SIVT, pp.325–330, 2010.
[8] W. Cheng, C. Hsieh, S. Lin, C. Wang, and J. Wu, "Robust algorithm for exemplar-based image inpainting," Proc. CGIV, pp.64–69, 2005.
[9] T. Kwok, H. Sheung, and C. Wang, "Fast query for exemplar-based image completion," IEEE Trans. Image Process., vol.19, no.12, pp.3106–3115, 2010.