PAPER

A Fast Exemplar-Based Image Inpainting Method Using Bounding Based on Mean and Standard Deviation of Patch Pixels

Jungmin SO^{†a)}, *Member* and Baeksop KIM^{†b)}, *Nonmember*

SUMMARY This paper proposes an algorithm for exemplar-based image inpainting, which produces the same result as that of Criminisi's original scheme but at the cost of much smaller computation cost. The idea is to compute mean and standard deviation of every patch in the image, and use the values to decide whether to carry out pixel by pixel comparison or not when searching for the best matching patch. Due to the missing pixels in the target patch, the same pixels in the candidate patch should be omitted when computing the distance between patches. Thus, we first compute the range of mean and standard deviation of a candidate patch with missing pixels, using the average and standard deviation of the entire patch. Then we use the range to determine if the pixel comparison should be conducted. Measurements with well-known images in the inpainting literature show that the algorithm can save significant amount of computation cost, without risking degradation of image quality.

key words: image inpainting, exemplar-based, bounding

1. Introduction

Image inpainting is a technique that fills in unknown regions in an image, so that the resulting image looks seamless to the human eye. Applications of image inpainting include recovering missing parts of an image, removing scratches and stains, and removing unwanted objects from the image and replacing it with the background. Image inpainting started to gain interest when Bertalmio et al. [5] proposed a PDE(Partial Differential Equation)-based method for recovering unknown parts of an image. Many different algorithms for image inpainting have been proposed since then, but the exemplar-based algorithm proposed by Criminisi [7] stands out in terms of quality and speed of computation.

In the exemplar-based inpainting, an image patch that contains unknown regions (the target patch) is compared with other patches (the candidate patches) in the entire image. Judging by the known pixels in the target patch, the most similar patch is selected, and the target patch is replaced with the candidate patch. The order of processing target patches is designed so that the structures are maintained when recovering the unknown region. Compared to PDE-based methods, exemplar-based methods can inpaint an image at a much faster speed. However, computation speed of image inpainting needs further improvement, in order to be used as a real-time function in small devices such

as mobile phones.

There has been efforts to improve the speed of exemplar-based inpainting. One example is to limit the search space based on the location of the target patch [1]. The idea is that since it is likely that the best candidate patch is found near the target patch, skip the candidate patches that are far away. The size of search space can be determined based on the image characteristics. However, depending on the image, the best matching patch can be outside the search region. In that case, the "quality" of the result may be degraded, although this quality metric may not be well-defined.

We propose an algorithm that improves speed of exemplar-based image inpainting, while retaining the same result as that of Criminisi algorithm. The main idea is to compute mean and standard deviation of image patches, and from these values obtain the lower bound on the patch distance. At run time, if the lower bound is greater than the current minimum patch distance, the patch is skipped. Since the patch distance is based on only the known pixels in the target patch and the corresponding pixels in the candidate patch, the range of mean and standard deviation of the known pixels are derived from the mean and standard deviation of the entire patch. Combined with pixel and patch ordering schemes proposed in [15], the proposed algorithm achieves up to 98% reduction in number of pixel comparisons, compared to the original Criminisi's method. When comparing with the scheme in [15], the proposed algorithm achieves up to 70% reduction in number of pixel comparisons.

The rest of the paper is organized as follows. In Sect. 2, previous works on inpainting are discussed. In Sect. 3, the original exemplar-based inpainting method proposed by Criminisi et al. is described, followed by the bounding and ordering algorithms to improve the computation time. In Sect. 4 the proposed patch pruning algorithm is explained, along with the proofs for obtaining lower bound on pixel distances. In Sect. 5, performance of the proposed algorithm is evaluated and compared with other methods, using seven images frequently used in inpainting literature. Finally, Sect. 6 concludes the paper.

2. Related Work

The initial approach for filling in missing regions in an image was to let surrounding pixels propagate into the unknown region. Since it is important to maintain structures,

Manuscript received November 6, 2014.

Manuscript revised March 13, 2015.

Manuscript publicized May 8, 2015.

[†]The authors are with the Dept. of Computer Engineering, Hallym University, Korea.

a) E-mail: jsol@hallym.ac.kr

b) E-mail: bskim@hallym.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2014EDP7369

partial differential equations (PDE) were used to achieve the goal [3]–[5], [11]. The problem with PDE-based solutions is that they take a very long time to complete, and also introduce blur in the resulting image.

Before Criminisi et al. introduced the exemplar-based inpainting, there were methods proposed to alleviate the drawbacks of PDE-based solutions. They include using statistical information such as gradient magnitude and relative gradient angle [18], choosing textures using tensor voting method [14], and iterative method using example-based synthesis [8]. Although these schemes claimed they are faster than PDE-based methods, the computation cost is much higher compared to the exemplar-based methods.

Criminisi et al. [7] proposed the exemplar-based inpainting method. Instead of using PDE, this method replaces an image patch with unknown pixels with the most similar image patch obtained from the known area of the image. Order of the target patches is carefully selected so that the structures are maintained. The exemplar-based method is much faster than PDE, and produces good results for both filling in small scratches and removing large unwanted objects. This method is the basis for our proposed algorithm, and is discussed in more detail in the next section.

After Criminisi's work, many efforts have been made to improve the exemplar-based inpainting method in terms of image quality and computation cost. One line of research focused on improving the quality of inpainted images [6], [9], [12], [16], [19], [20]. For example, Cheng et al. [6] proposed a new equation used to select the next patch in Criminisi's algorithm. Sun et al. [20] proposed a method where users specify the structures in the missing region. Komodakis et al. [16] formulated the patch selection process as a global optimization problem, compared to the greedy-style algorithm of Criminisi's. Fang et al. [9] proposed a method using multi-resolution approach. Hays et al. [12] proposed a method where candidate patches were selected from outside the target image, and Liu et al. [19] proposed a method where missing regions in the target is filled in using large displacement views of the same scene. While these works are able to improve quality of inpainted images, their computation cost is not reduced compared to Criminisi's, or in some cases the computation cost becomes higher, such as in methods where the candidate patch set becomes larger [12], [19].

The other line of research, and the focus of this paper, is on reducing computation cost of the exemplar-based method. The major bottleneck of Criminisi's exemplar-based method in terms of computation time is the time used for finding the most similar patch, and there were proposals to speed up the process. Anupam et al. [1] proposed to limit the search region based on the position of the target patch. The idea is based on the statistical observation that most similar patches are likely to be near the target patch. This scheme obviously decreases search time, but the quality of the resulting image can be degraded, if the best candidate patch is outside the search region. Barnes et al. [2] proposed a randomized algorithm for obtaining approximate

nearest neighbor. The algorithm is an iterative process, starting from a random guess. The iterative process consists of propagation and random search phase. In propagation phase the solutions are propagated to adjacent pixels using coherence, and in random search phase the current offset vector is perturbed by random offsets. Since it is an iterative algorithm, quality of the result depends on the number of iterations. This algorithm can significantly improve computation speed by limiting number of iterations. However, image quality is degraded when the number of iterations is small. Kwok et al. [17] proposed a novel technique of exemplar-based inpainting where the exemplars are decomposed into frequency coefficients and the most significant coefficients are selected for evaluating the matching score. Also, a local gradient-based algorithm is used to fill in the missing pixels in the query image block. The algorithm produces results at a significantly faster speed compared to Criminisi's method, and the quality of results is comparable. The problem with Kwok's method is that the image quality depends on the parameters, which needs to be set manually by the user. All these methods improve the computation time of the exemplar-based inpainting by sacrificing image quality, although one may claim that for some images reducing search space may produce comparable results or results that even look better. On the other hand, the proposed algorithm produces the same result as that of Criminisi's algorithm, while reducing significant amount of computation time.

3. Preliminaries

3.1 Exemplar-Based Inpainting

The main idea of the exemplar-based inpainting method [7] is to find an image patch within the whole image that will best fit the "target patch" that contains unknown pixels. An image patch is a two-dimensional block of pixels. The default patch size used in [7] and this paper is 9×9 . It is discussed in [7] that it is best to set the patch size as slightly larger than the largest distinguishable texture element.

Let us define Ω as the target region to be filled, and Φ as the source region that is to be used to fill in the target region. The source region, Φ , is the entire image I minus the target region Ω ($\Phi = I - \Omega$). The contour of the target region is called the "filling front", and is denoted as $\delta\Omega$. The inpainting algorithm first selects a target patch Ψ_p to fill in. (Ψ_p is a 9×9 patch centered at pixel p .) Then, a source patch Ψ_q is selected which has the most similar known region with the target patch. The source patch replaces the target patch, and this makes an iteration. The iterations continue until all the pixels in the unknown region are filled.

In an iteration, the first important part is to select the next target patch (Ψ_p) to be processed, which is said to be critical in preserving the structures. All target patches are ordered by a metric $P(p) = C(p)D(p)$, 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.

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

In the equations, $|\Psi_p|$ 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 $\nabla I_{n_p}^\perp$ is the isophote at p . Note that the confidence term is updated after every iteration.

Once the target patch is selected, the most similar source patch must be selected, which will replace the target patch. To do this, Ψ_p is compared to every patch centered at pixel q where $q \in \Phi$.

The metric used to measure the similarity of the two patches is called the “patch distance”. Patch distance is the sum of “pixel distances” of the known pixels in the target patch and corresponding pixels in the candidate patch. The pixel distance is the SSD (Sum of Squared Differences) of the pixel values. Suppose we use CIE Lab color space. Then, the pixel distance is defined as:

$$\delta_i = (c_{i,L} - c'_{i,L})^2 + (c_{i,a} - c'_{i,a})^2 + (c_{i,b} - c'_{i,b})^2 \quad (2)$$

where i is a pixel location in the patch, $c_{i,L}, c_{i,a}, c_{i,b}$ refer to the L, a, and b color values on one patch, and $c'_{i,L}, c'_{i,a}, c'_{i,b}$ refer to the L, a, and b values on the other patch. The *patch distance* between two patches Ψ_p and Ψ_q is defined as:

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

Note that only the known pixels are considered when computing patch distance. Once Ψ_p is compared to every candidate patch, the patch that has the minimum patch distance with Ψ_p is selected as the best matching patch, and it replaces Ψ_p to complete one iteration.

The process of comparing the target patch with every known patch is a very time-consuming job, which can take a very long time especially when executed on devices such as mobile phones. Many proposals have come up with ways to reduce computation time of exemplar-based inpainting as mentioned in the previous section. Here we consider schemes that produce the same result with the Criminisi’s algorithm. A simple bounding algorithm [10] combined with heuristics for pixel and patch ordering can reduce significant amount of computation cost while maintaining the same result.

3.2 Bounding and Ordering for Fast Patch Selection

To compute patch distance, pixel distance of corresponding pixels in the two patches is summed up for all known pixels. Since pixel distance is non-negative, we can stop calculating pixel distance if the current sum of pixel distances exceeds the minimum patch distance of already computed candidate patches. Algorithm 1 describes the process of bounding [15].

Now the performance of bounding algorithm is affected by the ordering of candidate patches and the pixels inside the patch. If the pixel distances and patch distances were already known, the best strategy would be to select the patch

Algorithm 1 Bounding algorithm for selecting the best patch

```

for each patch  $\Psi_q$  in  $(I - \Omega)$  do
   $D_{p,q} \leftarrow 0$ 
  for each pixel  $c_i$  in  $\Psi_p \cap (I - \Omega)$  do
     $\delta_i = (c_{i,L} - c'_{i,L})^2 + (c_{i,a} - c'_{i,a})^2 + (c_{i,b} - c'_{i,b})^2$ 
     $D_{p,q} \leftarrow D_{p,q} + \delta_i$ 
    if  $D_{p,q} > D_{p,q_B}$  then
      break
    end if
  end for
  if  $D_{p,q} < D_{p,q_B}$  then
     $q_B \leftarrow q$ 
     $D_{p,q_B} \leftarrow D_{p,q}$ 
  end if
end for

```

with the minimum patch distance first. Then, when computing patch distance of the target patch and a candidate patch, it is the best to order the pixels in ascending order of pixel distance. This way the number of pixel distance computation can be minimized.

Since the pixel and the patch distances are not known beforehand, heuristics can be used to order the patches and the pixels. In Kim et al. [15], candidate patches are ordered based on geographical distances between the candidate and the target patch. The strategy is based on the hypothesis that it is more likely to find the most similar patch near the target patch. When comparing two patches for patch distance, pixels are ordered based on the scarcity of its color (in gray scale). First, a histogram of gray-scale colors is generated for the entire image. When computing pixel distance, the algorithm selects the next pixel that is not been processed and has the lowest histogram value. The rationale behind this is that there is a negative correlation between frequency of colors in gray scale and the pixel distance. This is one of the heuristics that can be used to order pixels, and by no means optimal.

The patch and pixel ordering can push up the performance of the bounding algorithm, but still all candidate patches need to be compared with each target patch in order to select the best patch. The goal of the proposed method is to skip candidate patches that cannot be the best patch, without computing the pixel distances. Whether to skip or to process a candidate patch is determined based on mean and standard deviation of the patches, as described in the next section.

4. Patch Pruning Algorithm for Fast Patch Selection

4.1 Computing Lower Bound on Patch Distance

First of all, mean and standard deviation is computed for L, a, b of the pixels for every patch, of which all pixels are known. In other words, the following values are obtained for every known patch q : $\mu_L(q), \mu_a(q), \mu_b(q), \sigma_L(q), \sigma_a(q)$, and $\sigma_b(q)$.

Suppose we want to find a best patch that matches the

target patch Ψ_p . We select a candidate patch and compute the patch distance between two patches by summing up pixel distances. As the pixels are compared between two patches, patch distance increases monotonically. If the patch distance exceeds the previous minimum patch distance, then the computation process is stopped and we go to the next candidate patch.

In the proposed algorithm, before calculating pixel distances to get the patch distance, we first obtain the lower bound on patch distance between the two patches using mean and standard deviation of L , a , b values. If the lower bound exceeds the current minimum patch distance, the candidate patch is skipped.

To obtain lower bound, we use the lemma from [13] which says:

$$\text{dist}(x, y)^2 \geq d((\mu_x - \mu_y)^2 + (\sigma_x - \sigma_y)^2) \quad (4)$$

where d is the dimension of the vectors. In other words, the lower bound on the patch distance can be computed from mean and standard deviations of the two patches. For target patch p and candidate patch q :

$$D_{p,q}^2 \geq d \sum_{c \in \{L, a, b\}} ((\mu_c(p) - \mu_c(q))^2 + (\sigma_c(p) - \sigma_c(q))^2) \quad (5)$$

Even if we already obtained mean and standard deviation of candidate patches that are fully known, it cannot be directly used. Since some of the pixels are missing from the target patch, the pixel distance is computed using only the known pixels of the target patch, and the corresponding pixels of the candidate patch. Since the number and location of missing pixels are different for different target patches, the mean and standard deviation of candidate patches need to be computed every time a new target patch is processed. This is unacceptable in terms of cost. Instead, we compute the *range* of mean and standard deviation of the known pixels, from the mean and standard deviation of all pixels in the patch.

Suppose the number of pixels in a patch is d . In the case of a 9×9 patch, d is 81. For a target patch, suppose the number of known pixels is k . To find the best matching patch, one has to compute the pixel distance between the target patch p and the candidate patch q . Suppose X is the set of all pixels in the candidate patch, and $\tilde{X}$ is the set of pixels in the candidate patch that corresponds to the known pixels in the target patch. Then, the following lemma can be proved.

Lemma 1. *The mean and the standard deviation of $\tilde{X}$ can be bounded by following inequalities.*

$$\mu_c - \sqrt{\frac{d}{k}}\sigma_c \leq \tilde{\mu}_c \leq \mu_c + \sqrt{\frac{d}{k}}\sigma_c \quad (6)$$

$$0 \leq \tilde{\sigma}_c \leq \sqrt{\frac{d}{k}}\sigma_c \quad (7)$$

where μ_c and σ_c are the mean and standard deviation of X , $\tilde{\mu}_c$ and $\tilde{\sigma}_c$ are the mean and standard deviation of $\tilde{X}$, and

$c \in \{L, a, b\}$.

Proof. Note that for the proof, we just use μ and σ instead of μ_c and σ_c . First of all, since $\tilde{X} \subseteq X$,

$$\sum_{i \in \tilde{X}} (x_i - \mu)^2 \leq \sum_{i \in X} (x_i - \mu)^2 \quad (8)$$

Thus,

$$\frac{1}{k} \sum_{i \in \tilde{X}} (x_i - \mu)^2 \leq \frac{d}{k} \text{Var}(X) \quad (9)$$

Now, we can rewrite the left-hand side of Eq. (9).

$$\frac{1}{k} \sum_{i \in \tilde{X}} (x_i - \mu)^2 = \frac{1}{k} \sum_{i \in \tilde{X}} (x_i^2 - 2x_i\mu + \mu^2) \quad (10)$$

$$= \frac{1}{k} \sum_{i \in \tilde{X}} x_i^2 - 2\frac{1}{k}\mu \sum_{i \in \tilde{X}} x_i + \mu^2 \quad (11)$$

$$= E(\tilde{X}^2) - 2\tilde{\mu}\mu + \mu^2 \quad (12)$$

Since $\text{Var}(\tilde{X}) = E(\tilde{X}^2) - \tilde{\mu}^2$,

$$E(\tilde{X}^2) - 2\tilde{\mu}\mu + \mu^2 = \text{Var}(\tilde{X}) + \tilde{\mu}^2 - 2\tilde{\mu}\mu + \mu^2 \quad (13)$$

$$= \text{Var}(\tilde{X}) + (\tilde{\mu} - \mu)^2 \quad (14)$$

Finally, we obtain the following inequality.

$$\text{Var}(\tilde{X}) + (\tilde{\mu} - \mu)^2 \leq \frac{d}{k} \text{Var}(X) \quad (15)$$

Since variance and squared value are non-negative, we can bound the range of $\tilde{\mu}$ and $\tilde{\sigma}$ as follows.

$$\text{Var}(\tilde{X}) \leq \frac{d}{k} \text{Var}(X) \quad (16)$$

$$(\tilde{\mu} - \mu)^2 \leq \frac{d}{k} \text{Var}(X) \quad (17)$$

From Eqs. (16) and (17), we obtain Eqs. (6) and (7). $\square$

Now that we know the range of $\tilde{\mu}$ and $\tilde{\sigma}$, we can check to see if the candidate patch has possibility of becoming the best patch.

4.2 Pruning Patches Based on the Lower Bound

Now that we have the bound for mean and standard deviation of known pixels in the candidate patch, we can compute the bound on patch distance. Suppose $\tilde{\mu}_p$ and $\tilde{\sigma}_p$ are the mean and the standard deviation of known pixels in the target patch Ψ_p . Also, suppose μ_q and σ_q are the mean and the standard deviation of *all* pixels in the candidate patch. Remember that μ_q and σ_q are the values kept in record. We first obtain the minimum differences in mean and standard deviation:

$$\Delta\mu_{\min} = \begin{cases} \tilde{\mu}_p - (\mu_q + \sqrt{\frac{d}{k}}\sigma_q) & \text{if } \tilde{\mu}_p > \mu_q + \sqrt{\frac{d}{k}}\sigma_q; \\ (\mu_q - \sqrt{\frac{d}{k}}\sigma_q) - \tilde{\mu}_p & \text{if } \tilde{\mu}_p < \mu_q - \sqrt{\frac{d}{k}}\sigma_q; \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

$$\Delta\sigma_{min} = \begin{cases} \widetilde{\sigma}_p - \sqrt{\frac{d}{k}}\sigma_q & \text{if } \widetilde{\sigma}_p > \sqrt{\frac{d}{k}}\sigma_q; \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

Then, the lower bound on patch distance can be computed as

$$D_{p,q}^2 \geq d \sum_{c \in \{L,a,b\}} (\Delta\mu_{c,min}^2 + \Delta\sigma_{c,min}^2) \quad (20)$$

Suppose a target patch is selected to be replaced by a known patch. Initially, the minimum patch distance is set to a large number. Then, patch distance with candidate patches are computed. Before computing patch distance with a candidate patch, the lower bound on patch distance is computed based on mean and standard deviation, using Eq. (20). If the lower bound is larger than the current minimum patch distance, the patch is skipped without computing patch distance. If the lower bound is smaller than the current minimum, the patch distance is calculated. After computing patch distance, the minimum patch distance is updated if the new patch distance is smaller than the previous minimum. Since the skipped patches cannot have the minimum patch distance, the pruning does not change the result from the Criminisi's scheme.

Although patch pruning can reduce number of pixel distance calculations, additional computation cost is needed to calculate mean and standard deviation of every patch in the image. However, it is a procedure that needs to be done only once. It could also be done at offline, and mean and standard deviation of the patches can be store in the memory. To store mean and standard deviation of a patch, no more than 4 bytes are needed (2 bytes for mean and 2 bytes for standard deviation). If digits after the decimal point are cut off, we can store the information in 2 bytes, although it may slightly increase the search space due to rounding. Since mean and standard deviation should be computed for every patch in the image, the additional information will be of similar size as the original image. If memory is scarce, mean and standard deviation of the patches can be calculated once at the start of the inpainting procedure. When tested with the same hardware, computing mean and standard deviation of a patch took approximately 10 times the computation time of calculating pixel distance. So if we consider the cost of computing mean and standard deviation as 10, the computation cost of inpainting can be calculated for the case when mean and standard deviation is computed at online. For example, the cost of inpainting **Bungee** is 68.51 million pixel distance calculations when patch pruning is used, as shown in the next section. Accounting for cost of computing mean and standard deviation, this cost increases to 69.14, which is still much less than other schemes without patch pruning. Kwok et al. [17] also assumes that DCT transform, which requires larger amount of computation cost than computing mean and standard deviation, is done at offline. In the next section, offline calculation is assumed, therefore cost of computing mean and standard deviation as well as DCT transform is not included in the results.

5. Performance Evaluation

5.1 Comparison with Criminisi's Method and Its Improvements [15]

The experiments are conducted with seven images that are frequently used in inpainting literature. They are shown in Table 1.

The performance metric we use is the number of pixel distance calculations. All images were processed in CIE Lab color space, as it was done in [7]. The following methods are compared. The bounding and pixel/patch ordering methods are from [15], and Kwok's method is from [17]. Cost of computing mean and standard deviation is not included for the proposed method, assuming the computation is done offline as discussed in the previous section. Similarly the cost of DCT transform and gradient filling is not included for Kwok's method.

- Criminisi's: This is the original Criminisi's method without bounding.
- Bounding: The candidate patches are selected sequentially, starting from the patch at the top left corner. The pixels inside a patch is also selected in the same order. As the patch distance is calculated, if it exceeds the current minimum patch distance, the rest of the pixel distance calculations are skipped.
- Method A (Pixel ordering): Pixels inside a patch are compared in the ascending order of their frequency (histogram value) in the image.
- Method B (Patch ordering): Candidate patches are selected in the ascending order of geographical distance from the target patch.
- Method C (Patch pruning): The proposed patch pruning algorithm: patches are skipped if the lower bound on patch distance exceeds current minimum patch distance.
- Kwok's method: Patches are transformed using DCT transform, and m coefficients are selected to be included when computing patch distance.

The results are in Table 2, Fig. 1. The results shown in Fig. 1 are computation cost of schemes normalized to the Criminisi's algorithm.

In general, we can see that the patch pruning algorithm significantly reduces the computation cost of exemplar-based inpainting. For **Rein River**, the pruning method combined with bounding and ordering can reduce 98% of the computation cost from the Criminisi's original method. When comparing with bounding and ordering, pruning reduces 67% of pixel distance computations. For all images used in the experiments, the patch pruning algorithm reduces 20% to 70% of computation cost from the case where only bounding and ordering algorithms are used. As shown in the result, the ordering algorithms produce significant improvements in some images but show limited impact on

Table 1 The images used for performance evaluation. Images in the second column are the results from Criminisi’s original inpainting method. Images in the third column are the results from the proposed method. Images in the fourth column are the results from Kwok’s method [17] where m is 20 and r is 1%.

Title	Marked for inpainting	Criminisi [7]	Proposed method	[17] (m=20, r=1.0%)
Rein River
Rice Field
Woman
Pumpkin
Golf
Bungee
Cable Car

Table 2 Computation cost in terms of number of pixel distance calculations for various schemes. Units are in millions.

Title	Rein River	Rice Field	Woman	Pumpkin	Golf	Bungee	Cable Car
Size	500 × 375	500 × 334	360 × 480	472 × 332	350 × 262	206 × 308	342 × 450
Criminisi's	1057.49	1095.67	10973.86	2782.09	1599.34	1046.94	1558.60
Bounding	113.64	227.33	1262.84	369.89	222.61	198.49	325.28
Method A	100.74	157.48	1021.74	349.46	188.79	121.67	226.71
Method B	81.99	200.54	1250.90	255.84	196.79	181.27	321.60
Method C	42.92	125.62	728.42	197.80	150.12	101.83	259.16
Method A+B	71.72	138.72	1005.33	240.48	164.14	109.47	222.78
Method A+C	34.23	91.02	585.36	182.13	121.94	74.52	179.82
Method B+C	31.87	110.71	728.69	112.14	130.90	94.43	256.15
Method A+B+C	23.90	79.09	579.53	101.37	104.50	68.51	176.50
[17] (m=20,r=1%)	71.26	113.22	780.56	248.84	153.48	96.65	125.28

Fig. 1 Computation cost normalized to the Criminisi's algorithm.

some images. For images such as **Rein River** and **Pumpkin**, the pixel ordering only reduces 12% and 6% of the computation cost, respectively. For **Bungee**, **Cable Car** and **Woman**, the patch ordering algorithm reduces 10%, 2% and 1% of the computation cost, respectively. However, the patch pruning algorithm reduces more than 20% computation cost for all tested images, from what is already achieved by bounding and ordering algorithms. Table 2 also shows the performance of patch pruning when it is not used with the ordering schemes (Method C). Although the amount of benefit varies with images, the results show that patch pruning alone can significantly reduce computation cost. For **Rein River**, patch pruning requires 43 million pixel distance calculations when pixel ordering and patch ordering combined needs 72 million calculations. When all three schemes are used together, the number is brought down to 24 million pixel distance calculations.

5.2 Comparison with Kwok's Method [17]

Kwok et al. [17] uses discrete cosine transform (DCT) in order to reduce time searching for the best matching patch. The method consists of three processes. First, the unknown pixels in the target patch are filled using gradient-based filling method. Second, the target patch and the candidate patches are transformed using DCT. Then, only m significant elements are left and other values are truncated to zero.

Fig. 2 The inpainted result of **Bungee**. (a) shows the result of the proposed scheme, while other images are results of Kwok's method [17]. The number of pixel distance calculations for (a)-(f) are 68.51, 71.28, 103.26, 38.43, 96.65, and 156.55 millions, respectively.

The m elements are used to compute the patch distance between the target patch and the candidate patches. Using this method, 0.1% of the candidate patches are selected as best patches. Finally, a single patch is selected from the best patches using the standard SSD similar to the Criminisi's method. Here m and 0.1% are the parameters that control the computation cost and the image quality. In [17], it is not reported how m is set, and 0.1% is always used for selection of the best patches.

Figure 2 shows the result of running Kwok's method on **Bungee** with various parameters. Figure 2(b)-(f) shows the resulting images when the m and r parameters are (20, 0.1), (40, 0.1), (5, 1.0), (20, 1.0), and (40, 1.0) respectively, where r is the percentage of patches selected for SSD calculations.

Computation costs in terms of pixel distance calculations for the five parameter sets are 71.28, 103.26, 38.43, 96.65, and 156.55 millions, respectively. When r is 0.1%, the structure at the center of the image is not recovered well. Image quality is comparable to the result of proposed method in Fig. 2(a) when r is 1%, and m is higher than 20. The computation cost of Fig. 2(e) is 96.65, which is higher than the proposed method. Note that bounding algorithm is also applied to Kwok's method for fair comparison.

In Table 1, images in the fourth column are the result of Kwok's method, where m is 20 and r is 1%. The parameters are chosen so that the computation cost is similar to the proposed method. For **RiceField**, **Bungee**, and **Cable Car**, the quality of results are comparable. For **ReinRiver**, **Woman**, and **Golf**, quality of the proposed algorithm is better, even though the computation cost is lower as shown in Table 2. For **Pumpkin**, both schemes fail to produce good results. In terms of computation cost, Kwok's scheme achieves lower computation cost for **Cable Car**, while for other images the proposed scheme shows lower computation cost.

In summary, performance of Kwok's method depends on images. For some images, the scheme produces good image quality at low computation cost. For other images, quality is severely degraded. However, the proposed scheme retains the result of Criminisi's scheme while reducing significant amount of computation cost.

Finally, a significant advantage of patch pruning is that it can be used with any method that uses patch distance calculations. The works of Anupam et al. [1], Barnes et al. [2], and Kwok et al. [17] all use patch distance calculations. For Anupam's method, patch pruning can be used after selecting the search space. For Barnes' method, patch pruning can help quickly skip patches when trying to find a good patch with propagation and random search. For Kwok's method, patch pruning can be applied to DCT-transformed patches. If the system has enough additional memory to store mean and standard deviation of patches in the image, patch pruning can always be applied to reduce computation cost without affecting the end result.

6. Conclusion

A patch pruning algorithm is proposed to speed up the process of exemplar-based inpainting. The lower bound on patch distance can be obtained from mean and standard deviation of pixel values of the two patches. However, since only known pixels in the target patch must be compared with the candidate patch, the mean and standard deviation of all the pixels cannot be directly used. Instead, we obtain a range of mean and standard deviation for the known pixels, from the mean and standard deviation of all pixels in the patch. Using this range, we can obtain the lower bound on pixel distance without computing mean and standard deviation of the known pixels in the candidate patches for every target patch. Experiments show that the proposed algorithm significantly reduces computation cost, and can be used with bounding and ordering algorithms to reduce com-

putation cost further. This result pushes up the practicality of exemplar-based inpainting to be used as a real-time image processing feature on small customer devices such as mobile phones.

Acknowledgements

This research was supported by Hallym University Research Fund 2013 (HRF-G-2013-5), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2054526).

References

- [1] Anupam, P. Goyal, and S. Diwakar, "Fast and Enhanced Algorithm for Exemplar Based Image Inpainting," 2010 Fourth Pacific-Rim Symposium on Image and Video Technology, pp.325–330, 2010.
- [2] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman, "Patch-match: a randomized correspondence algorithm for structural image editing," ACM SIGGRAPH 2009 papers on - SIGGRAPH '09 (2009)
- [3] M. Bertalmio, "Strong-continuation, contrast-invariant inpainting with a third-order optimal PDE," IEEE Trans. Image Process., vol.15, no.7, pp.1934–1938, 2006.
- [4] M. Bertalmio, A.L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," Proc. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, pp.I-355–I-362, 2001.
- [5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," Proc. 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00, pp.417–424, 2000.
- [6] W. Cheng, C. Hsieh, S. Lin, C. Wang, and J. Wu, "Robust algorithm for exemplar-based image inpainting," CGIV, pp.64–69, 2005.
- [7] A. Criminisi, P. Perez, and K. Toyama, "Region Filling and Object Removal by Exemplar-Based Image Inpainting," IEEE Trans. Image Process., vol.13, no.9, pp.1200–1212, 2004.
- [8] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," ACM Trans. Graphic., vol.22, no.3, pp.303–312, 2003.
- [9] C.-W. Fang and J.-J. Lien, "Rapid Image Completion System Using Multiresolution Patch-Based Directional and Nondirectional Approaches," IEEE Trans. Image Process., vol.18, no.12, pp.2769–2779, 2009.
- [10] K. Fukunaga and P.M. Narendra, "A Branch and Bound Algorithm for Computing k-Nearest Neighbors," IEEE Trans. Comput., vol.C-24, no.7, pp.750–753, 1975.
- [11] T. Georgiev, "Photoshop healing brush: a tool for seamless cloning," ECCV, pp.1–8, 2004.
- [12] J. Hays and A.A. Efros, "Scene completion using millions of photographs," ACM Trans. Graphic., vol.26, no.4, 2007.
- [13] Y. Hwang, B. Han, and H.-K. Ahn, "A fast nearest neighbor search algorithm by nonlinear embedding," 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp.3053–3060, 2012.
- [14] J. Jia and C.-K. Tang, "Image repairing: robust image synthesis by adaptive ND tensor voting," 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., pp.I-643–I-650, 2003.
- [15] B. Kim, J. Kim, and J. So, "Pixel and Patch Reordering for Fast Patch Selection in Exemplar-Based Image Inpainting," IEICE Trans. Inf. & Syst., vol.E96-D, no.12, pp.2892–2895, 2013.
- [16] N. Komodakis and G. Tziritas, "Image Completion Using Global Optimization," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06), pp.442–452, 2006.

- [17] T.-H. Kwok, H. Sheung, and C.C.L. Wang, "Fast Query for Exemplar-Based Image Completion," *IEEE Trans. Image Process.*, vol.19, no.12, pp.3106–3115, 2010.
- [18] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," *Proc. Ninth IEEE International Conference on Computer Vision*, vol.1, pp.305–312, 2003.
- [19] C. Liu, Y. Guo, L. Pan, Q. Peng, and F. Zhang, "Image completion based on views of large displacement," *The Visual Computer*, vol.23, no.9-11, pp.833–841, 2007.
- [20] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM Trans. Graphic.*, vol.24, no.3, pp.861–868, 2005.

Jungmin So received the B.S. degree in Computer Engineering from Seoul National University in 2001, and Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign in 2006. He is currently an assistant professor in Department of Computer Engineering, Hallym University. His research interests include mobile computing and image processing.

Baeksop Kim received the B.S. degree in Electronic Engineering from Hanyang University in 1978, and he received Masters and Ph.D. degree in Electronic Engineering from Korea Advanced Institute of Science and Technology in 1980 and 1985, respectively. He is currently a professor in Department of Computer Engineering, Hallym University in Korea. His research interests include pattern recognition, image processing and computer vision.