Fast Query for Exemplar-Based Image Completion

Tsz-Ho Kwok, Hoi Sheung and Charlie C.L. Wang*, *Member, IEEE*

***Abstract*—In this paper, we present a fast algorithm for filling unknown regions in an image using the strategy of exemplar-matching. Unlike the original exemplar-based method using exhaustive search, we decompose exemplars into the frequency coefficients and select fewer coefficients which are the most significant to evaluate the matching score. We have also developed a local gradient-based algorithm to fill the unknown pixels in a query image block. These two techniques bring the ability of input with varied dimensions to the fast query of similar image exemplars. The fast query is based on a search-array data structure, and can be conducted very efficiently. Moreover, the evaluation of search-arrays runs in parallel maps well on the modern graphics hardware with *Graphics Processing Units* (GPU). The functionality of the approach has been demonstrated by experimental results on real photographs.**

***Index Terms*—TEC-RST: Image completion, exemplar, DCT, parallel, GPU**

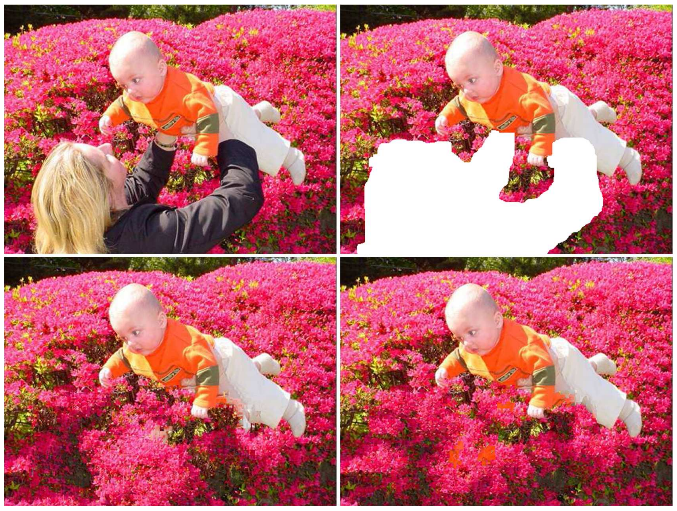


Fig. 1. Our fast query method speeds up the exemplar-based inpainting method. For the input image with 538 $\times$ 403 pixels (top-left), we can successfully fill the large removed region (about 27.15% of the whole image has been blanked out) in 11.56 seconds. Our result is shown in bottom-right. The algorithm in [2] takes about 133 seconds even after the ANN library [7] has been employed to compute the best matched candidate exemplar(bottom-left).

I. Introduction

The removal of objects or the recovery of damaged portion in a given image, known as *image completion*, is an important task in photo editing and video post-processing. Specifically, given an input image I with a missing or unknown region Ω, we want to propagate structure and texture information from the known region $I \setminus \Omega$ to Ω. Most of the existing methods in literature take quite a long time to retouch one image, which is far from a practical use in an interactive image processing and editing. The Healing Brush tool in the commercial software Adobe Photoshop is based on solving a *Partial Differential Equation* (PDE) in the unknown region with the Dirichlet boundary condition [1]. Although it can reach the interactive speed, it does not give satisfactory results in the regions with large unknown areas and highly textured regions. Therefore, Criminisi et al. [2] developed the exemplar-based image inpainting method to overcome these defects. However, as the exemplar-based method needs to repeatedly search the best matched exemplar to fill the unknown regions, it takes about tens or even hundreds of seconds to process an image with a moderate resolution (e.g., the photograph in Fig.1). Different from the image query in texture synthesis, the input of query in image completion [2] has varied dimensions during the region filling procedure. We present a fast query method in this paper to speed up the exemplar-based image inpainting so that it can speed up the computation on a consumer level PC equipped with modern graphics hardware.

In literature, many techniques have been developed. They can be roughly classified into PDE-based, exemplar-based and statistical-based. PDE-based method in [3], [4] diffuses the known pixels into the missing regions and smoothly propagates image information from the surrounding areas along the isophotes direction. The recovered results are highly smooth. This method works well for small damaged regions. However, when the reconstructed area is large, it gives an unreal blurry artifact that lacks texture. Drori et al. [5] incorporated pyramid image approximation and adaptive image fragment to achieve impressive results. Nevertheless, all these approaches are extremely slow due to the high computational complexity. Statistical-based method [6] uses the context of statistical learning to fill the missing regions but it also fails in highly textured photographs and takes a long time to compute the results.

Recently, more and more researchers have started to consider using exemplar-based methods to complete images with a large portion removed, which are in fact a combination of texture synthesis [8] and inpainting. The first attempt to use exemplar-based synthesis for image completion was made by Harrison [9] who filled the pixels in the target region by the level of "textureness" on the neighborhoods of a pixel. Although the intention sounds good, strong linear structures are often overruled by nearby noises in his approach. Jia and Tang [10] presented a technique for filling image regions by explicitly segmenting the unknown area into different homogeneous texture areas with a tensor voting method. However, their approach requires both an expensive segmentation step and a difficult choice making on how to stitch the boundary be-

*Corresponding Author. The authors are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, China (Tel: (852)26098337 / Fax: (852) 26036002) (e-mail: thkwok@mae.cuhk.edu.hk, hsheung@mea.cuhk.edu.hk, cwang@mae.cuhk.edu.hk).

tween two textures. Criminisi et al. [2] proposed an exemplar-based image completion algorithm. Their method computes the priorities of patches to be synthesized through a best-first greedy strategy which depends on the priority assigned to each patch on the filling-front, where the patch filling order is determined by the angle between the isophote direction and the normal direction of the local filling front. This algorithm works well in large missing regions and textured regions. Our proposed algorithm fills the missing region in a similar way but with a more efficient matching (query) technique. Several variants were proposed after the approach in [2] was presented. Priority-BP (*BP* stands for *belief propagation*) was posed in the form of a discrete global optimization problem in [11]. The priority-BP was introduced to avoid visually inconsistent results; however, it takes a longer computing time than [2] and needs user guidance. Sun et al. [12] introduced a novel structure propagation approach to image completion, where the user manually specifies the important missing structures by extending a few curves or line segments from a known region to the unknown. Their approach synthesizes image patches along these user-specified curves using patches selected around the curves in the known region. Structure propagation is formulated as a global optimization problem by enforcing structure and consistency constraints, which is also very time-consuming when working on large images.

Retouching an image using other resources from database or Internet is a new strategy which was first researched by [13]. The recent approaches include the usage of large displacement views in completion [14] and the image database in re-coloring [15]. However, when there is a large number of candidate patches, the processing time will become even longer. Here we focus on the speed-up problem of image completion using a fast query method which to the best of our knowledge has not been discussed before.

Some recent researches make the effort on speeding up the time-consuming image completion procedure. Fang et al. [16] presented an image completion algorithm using multi-resolution patch-based directional and non-directional approach. Although the speed has been improved than [2], it is slower than ours. For example, the photograph shown in Fig.1, their method needs 35 seconds to complete. Their approach spends a lot of time on the training step. Barnes et al. presented the PatchMatch method, a randomized correspondence algorithm, for structural image editing in [17]. Their approach is dramatically fast. However, when being applied on the regions with structural information (e.g., sharp edges), their results in image completion is not as good as [2] and ours. The guidance given by sketch input is need in their examples have such structures. The randomized correspondence algorithm needs the input query have a fixed dimension, so that it cannot be applied directly in the procedure of [2].

A. Exemplar-based image inpainting

The procedure of exemplar-based image inpainting [2] is briefed below. Each pixel $\mathbf{p}$ on the given image I has a confidence term such that, during initialization, $C(\mathbf{q})$ is assigned to $C(\mathbf{p}) = 1$ $(\forall \mathbf{p} \in \Omega)$ and $C(\mathbf{p}) = 0$ $(\forall \mathbf{p} \in I \setminus \Omega)$. After extracting the manually selected initial front $\partial\Omega^0$, the algorithm repeats the following steps until all pixels in Ω have been filled.

1) Identify the filling front $\partial\Omega^t$, and exit if $\partial\Omega^t = \emptyset$.
2) Compute (or update) the priorities of every pixel $\mathbf{p}$ on the filling front $\partial\Omega^t$ by $P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p})$ defined in an image block $\Psi_\mathbf{p}$ centered at $\mathbf{p}$. $C(\mathbf{p})$ and $D(\mathbf{p})$ are the confidence term and the data term respectively defined as
$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_\mathbf{p}\cap\Omega} C(\mathbf{q})}{|\Psi_\mathbf{p}|},\ D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{n_p}}^{\perp} \cdot \mathbf{n_p}|}{\alpha} \quad (1)$$
where $|\Psi_\mathbf{p}|$ is the area of $\Psi_\mathbf{p}$, α is the normalization factor (e.g., $\alpha = 255$ for a typical grey-level image), $\mathbf{n_p}$ is the unit normal vector which is orthogonal to the front $\partial\Omega$ at $\mathbf{p}$, and $\nabla I_{\mathbf{n_p}}^{\perp}$ is the isophote at $\mathbf{p}$.
3) Find the patch $\Psi_{\hat{\mathbf{p}}}$ which has the maximum priority among all patches centered at the filling front $\partial\Omega^t$.
4) Find the exemplar patch $\Psi_{\hat{\mathbf{q}}}$ in the filled region which best matches the query image block $\Psi_{\hat{\mathbf{p}}}$.
5) Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}$ $(\forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega)$.
6) Update $C(\mathbf{p}) = C(\hat{\mathbf{p}})$ $(\forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega)$.

Note that, in step 2, the pixel which is surrounded by more confident (known) pixels and is more likely to let the isophote flow in has a higher priority. To efficiently find a patch $\Psi_{\hat{\mathbf{p}}}$ having the maximum priority in step 3, we keep a maximum heap during the region filling, which can be locally updated after filling each image block in step 5 and 6. In the original algorithm proposed by Criminisi et al. [2], the best matched image block $\Psi_{\hat{\mathbf{q}}}$ (in step 4) is a patch in the filled region which minimizes the *Sum of Squared Differences* (SSD) between $\Psi_{\hat{\mathbf{p}}}$ and $\Psi_{\hat{\mathbf{q}}}$ (i.e., $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$) defined on those already filled pixels. As explained in [2], $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$ is evaluated in the CIE Lab color space because of its property of perceptual uniformity.

In the routine of exemplar-based image inpainting, the most time consuming step is finding the best matched patch $\Psi_{\hat{\mathbf{q}}} = \arg\min_{\Psi_{\hat{\mathbf{q}}}\in I\setminus\Omega} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$. Here, one must conduct an image query with varied dimension on input which foils many of the existing acceleration schemes (e.g., the KD-tree based *Approximate Nearest Neighbor* (ANN) search in [7]). Another acceleration scheme – the *Tree Structured Vector Quantization* (TSVQ) in [8] can be used; however, it gives poorer results since the searching result does not guarantee to be the best matched one. Also, the construction time of binary tree in TSVQ is quite long, which makes it not as efficient as our approach presented in this paper.

B. Main Contribution

We developed a fast query method for speeding up the exemplar-based image completion. For the evaluation of matching score, we decompose exemplars into frequency domain using *Discrete Cosine Transformation* (DCT) and determine the best matched patches using fewer coefficients more efficiently with the help of the search-array data structure. In order to compute DCT on patches with unknown pixels, their pixel values are determined by a local gradient-based filling, which can preserve the continuity of image information better than simply filling unknown pixels by the average color of known pixels. Our approach can greatly reduce the computing time of exemplar-based image completion from hundreds of seconds to a few seconds. This is an important feature which makes this image editing tool practically usable.

II. Image Query in Transformed Domain

Using exhaustive search to determine the best matched patch which minimizes SSD in the above algorithm can hardly reach the speed reported in [2]. Therefore, we tried to employ the efficient *Approximate Nearest Neighbor* (ANN) library [7] which is implemented by KD-tree to search for the best matched patch $\Psi_{\hat{\mathbf{q}}}$ for a given query patch $\Psi_{\hat{\mathbf{p}}}$ at the filling front $\partial\Omega$, where every image block is considered as a high-dimensional point in the feature space and the distance in such a feature space is equal to the SSD. However, the unknown pixels in $\Psi_{\hat{\mathbf{p}}}$ change from time to time, thus the dimension cannot be fixed. Since no description about it has been given in [2], we used the average color to fill the unknown pixels so that the dimension of KD-tree is fixed. Surprisingly, same results are obtained with similar lengths of time – we tested our implementation in this way on Fig.13 and 15 in [2].

Further study shows that when increasing the dimension of KD-tree from 20 to more than 200 (e.g., a window size with 9×9 pixels is a point in the space with a dimension of $9 \times 9 \times 3 = 243$), the performance of KD-tree drops significantly. Thus, in order to improve the efficiency, we need a method to approximate the image block using fewer coefficients than all pixels color value. Naively selecting several pixels from an image block can never successfully present (and reconstruct) the information in the whole image block. We therefore consider this problem in the transformed domain of images.

A. Comparison of using different transformations

The idea of using fewer m coefficients to approximate a given $n \times n$ image (i.e., $m \ll n^2$) are stimulated by the image compression techniques in [18] and the image query metric in [19]. Therefore, *Principal Component Analysis* (PCA) [20], [21], *Fast Fourier Transform* (FFT) [22], *Discrete Cosine Transformation* (DCT), standard and non-standard Haar wavelet decomposition [23], [24] are considered as candidate transformation strategies for our fast query.

Principal Component Analysis (PCA) is powerful in dimensional reduction as it allows us to discard vectors with insignificant variance, thus it is widely used in data analysis. It works well in human body modeling [25]. Different body parameters highly relate to each other, such as bodies with larger hip girth than the average are likely to have larger thighs. PCA is very useful in dealing with these kinds of redundancy. Nevertheless, there may not be such kinds of redundancy in image blocks. Take a highly textured 8×8 image block shown below as an example.

$$\begin{bmatrix} 19 & 179 & 94 & 67 & 195 & 64 & 2 & 208 \\ 180 & 237 & 119 & 147 & 59 & 242 & 95 & 249 \\ 60 & 175 & 128 & 223 & 149 & 76 & 229 & 219 \\ 101 & 144 & 231 & 15 & 117 & 40 & 81 & 21 \\ 68 & 97 & 52 & 112 & 219 & 92 & 152 & 86 \\ 211 & 161 & 86 & 21 & 168 & 188 & 76 & 60 \\ 253 & 92 & 146 & 143 & 90 & 179 & 32 & 81 \\ 165 & 104 & 124 & 137 & 88 & 178 & 99 & 250 \end{bmatrix}$$

After PCA, the vector containing the percentage of the total variance explained by each principal component is as follows:

$$\{\ 35.48 \quad 32.80 \quad 13.48 \quad 10.47 \quad 5.66 \quad 1.77 \quad 0.34 \quad 0\ \}$$

If we want to keep 90% of accuracy, at least 4 vectors, i.e. 32 coefficients in total are needed to be retained, which means 50% of coefficients must be kept. In this case, the efficiency of reduction is quite low. Since our target is to keep 10% or even fewer coefficients of the whole image block, PCA is out of our consideration.

FFT is not considered as well because the imaginary part is not useful in image querying. Similar to other multimedia applications, such as a compact version of FFT, DCT is more appropriate than FFT in image querying since its computation is simpler and faster in general. Here, we consider the two-dimensional normalized type-II DCT which is widely used in image compression (e.g., JPEG) and select the standard and non-standard Haar wavelet decomposition as candidate transformations because they are all linear and their basic functions are orthonormal.

Lemma If the basic functions of transformation are orthonormal, squared L^2 error of the transformed coefficient differences between two image blocks $\Psi_{\mathbf{p}}$ and $\Psi_{\mathbf{q}}$ are the same as the squared L^2-norm difference between two images on pixel values.
Proof. See Appendix.

From the above lemma, we know that the difference between image blocks can be measured by the difference in their transformed coefficients in DCT, the standard and non-standard Haar wavelet decomposition. Now the problem left is that which transformation can reconstruct the original image with the fewest number of coefficients – especially in highly textured regions.

Our first try is to use m coefficients at the upper-left corner of the transformed image block. In DCT and wavelet transformations, these coefficients are the lower frequency components. The heuristic to do this is based on the reason that human eyes are more sensitive to noises in low frequency components than in high frequency ones [26]. Other coefficients are truncated to zero. However, the images reconstructed from these retained coefficients using Inverse DCT and the standard and non-standard Haar wavelet reconstruction are far from the input when working on highly textured images (as the example texture images shown in Fig.2). In other words, it is not easy to use these m coefficients to distinguish the original textured images from the reconstructed ones.

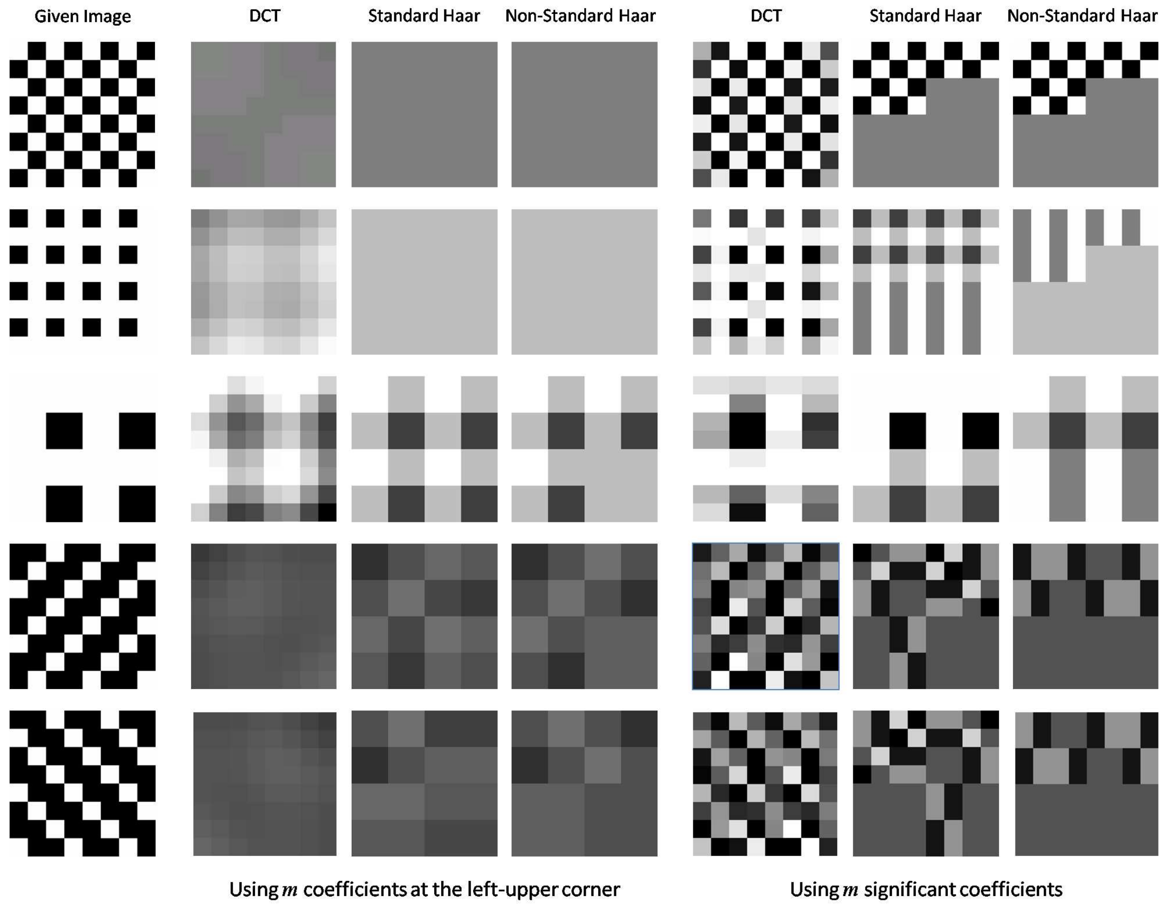

Given Image
DCT
Standard Haar
Non-Standard Haar
DCT
Standard Haar
Non-Standard Haar
Using m coefficients at the left-upper corner
Using m significant coefficients

Fig. 2. Study the query performance of m selected coefficients in the transformed domain by their ability to reconstruct the highly textured input image – the better the original image is reconstructed, the more accurate the image difference error is reflected by these m coefficients. Using m significant coefficients on DCT gives the best performance.

To improve the performance on textured images, we use m significant coefficients (i.e., the m coefficients with the largest magnitude) in the second test. After checking the reconstruction performance (which in fact measures the query performance) on the same set of textured images in the right of Fig.2, it is not difficult to find that using m significant coefficients on DCT works best. Therefore, we conducted the strategy of using m significant DCT coefficients in our fast query algorithm.

B. Gradient-based filling

For computing DCT coefficients on image blocks with unknown pixels, if the unknown pixels are filled with average color of known pixels, the DCT coefficients do not reflect the texture or the structural information in the block very well. For example, if the missing region Ω is located at the center of an image block with progressive color change from left to right, filling Ω with average color is not a good approximation. For a smooth image, the gradient at pixels will be approximately equal to zero. Based on this observation, we developed a gradient-based filling method to determine the unknown pixels before computing DCT.

In detail, for each unknown pixel $\mathbf{p}_{i,j}$, letting the discrete gradient at this pixel be zero will lead to linear equations

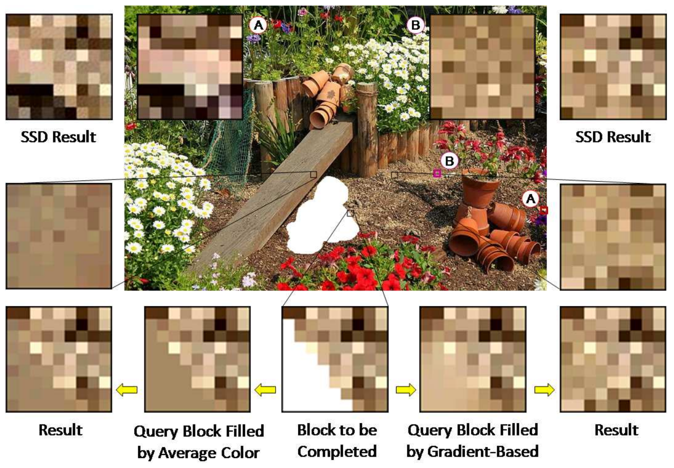

A
B
B
A
SSD Result
SSD Result
Result
Query Block Filled by Average Color
Block to be Completed
Query Block Filled by Gradient-Based
Result

Fig. 3. Filling unknown pixels by (left) the average color of known pixels vs. (right) the gradient-based filling. Top four image blocks shows the query and filled results using SSD based metric, where (A) is the query result by filling unknown pixels by the average color and (B) is by the gradient-based filling. Bottom left and bottom right are the query and filled results using the truncated m-dominant DCT coefficients.

relating $\mathbf{p}_{i,j}$ to its left/right and top/bottom neighbors – $\mathbf{p}_{i\pm1,j}$ and $\mathbf{p}_{i,j\pm1}$ – using the forward difference or the backward difference. Therefore, for l unknown pixels, we can have

k linear equations with $k > l$ which is actually an over-determined linear system. The optimal values of unknown pixels can then be computed by the Least-Square solution which minimizes the norm of gradients at unknown pixels. Figure 3 gives a comparison of an image block with unknown pixels filled by average color versus gradient-based filling. The queries are taken by using both the conventional SSD and the truncated m-dominant DCT coefficients (i.e., method presented in above section). It is not difficult to find that a better matched exemplar image block is found when filling the unknown pixels in a query image block using the gradient-based method. More specifically, the matched image blocks given by an input with gradient-based filling usually have better compatibility across the boundary between the known and the unknown pixels.

Note that, the gradient filling may generate smoother images at those unknown pixels when the known pixels are highly textured. This will be overcome by a revision of the Criminisi et's method [2]. We find more than one matched patches (actually 0.1% of the toal exemplars), among which the one with the highest SSD score on the known pixels is finally selected. Furthermore, using m-dominant DCT coefficients in selection will also reduce the effect led by the pixels filled in this step.

III. Fast Image Query Algorithm

We developed a fast query algorithm to search among a set of $\hat{m}$ exemplar patches $\Psi_{\mathbf{q}_k}$ $(k = 0, \cdots, \hat{m}-1)$ in the filled region to find the best matched exemplar patch $\Psi_{\hat{\mathbf{q}}}$ for the query image block $\Psi_{\hat{\mathbf{p}}}$. More specifically, we need to find

$$\Psi_{\hat{\mathbf{q}}}^{M} = \arg\min_{\Psi_{\hat{\mathbf{q}}} \in I \setminus \Omega} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}}) \tag{2}$$

with $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$ being the *Sum of Squared Differences* (SSD) which is actually the squared L^2 norm as

$$d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}}) = \sum_{(i,j)} \|\mathbf{p}_{i,j} - \mathbf{q}_{i,j}\|^2 \ \ (\forall \mathbf{p}_{i,j} \in \Psi_{\hat{\mathbf{p}}}, \mathbf{q}_{i,j} \in \Psi_{\hat{\mathbf{q}}}) \tag{3}$$

where the color at each pixel $\mathbf{p}_{i,j}$ (and $\mathbf{q}_{i,j}$) is a vector of three components in CIE Lab color space.

As aforementioned, we transform the exemplar image blocks and the query image block into their frequency domain by DCT. Then, only m significant coefficients are retained for each color channel of every image while other coefficients are truncated to zero. In the rest of this paper, we use $\hat{\mathbf{p}}_{i,j}^{t}$ to denote the truncated DCT coefficient at (i,j) in the image block $\Psi_{\hat{\mathbf{p}}}$ for the color channel t $(t = L, a, b)$. The positions of m significant coefficients change during the image completion routine. Therefore, if we simply use ANN to find the best matched image $\Psi_{\hat{\mathbf{q}}}$ of the query image block $\Psi_{\hat{\mathbf{p}}}$ by the SSD on truncated DCT coefficients as

$$\tilde{d}(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}}) = \sum_{t} \sum_{(i,j)} (\hat{\mathbf{p}}_{i,j}^{t} - \hat{\mathbf{q}}_{i,j}^{t})^2, \tag{4}$$

we still need to have the KD-tree of n^2 dimensions – the computation cannot be sped up.

A. Efficient Scoring Images

To avoid evaluating the Euclidean distance in the n^2-dimension space, we first simplify Eq.(4) and then employ the search array data-structure to evaluate $d^t(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$ more efficiently.

Expanding Eq.(4) and eliminating the zero terms, we have

$$\tilde{d}(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}}) = \tilde{d}_{\hat{\mathbf{p}}} + \tilde{d}_{\hat{\mathbf{q}}} - 2\tilde{d}_{\hat{\mathbf{p}}\hat{\mathbf{q}}} \tag{5}$$

with

$$\begin{aligned} \tilde{d}_{\hat{\mathbf{p}}} &= \sum_{t} \sum_{\hat{\mathbf{p}}_{i,j}^{t} \neq 0} (\hat{\mathbf{p}}_{i,j}^{t})^2, \\ \tilde{d}_{\hat{\mathbf{q}}} &= \sum_{t} \sum_{\hat{\mathbf{q}}_{i,j}^{t} \neq 0} (\hat{\mathbf{q}}_{i,j}^{t})^2, \\ \tilde{d}_{\hat{\mathbf{p}}\hat{\mathbf{q}}} &= \sum_{t} \sum_{\hat{\mathbf{q}}_{i,j}^{t} \neq 0,\ \hat{\mathbf{p}}_{i,j}^{t} \neq 0} \hat{\mathbf{q}}_{i,j}^{t} \hat{\mathbf{p}}_{i,j}^{t}. \end{aligned}$$

In practice, the value of $\tilde{d}_{\hat{\mathbf{p}}}$ for a given query image block will not change; therefore, the metric for scoring the similarity between $\Psi_{\hat{\mathbf{p}}}$ and $\Psi_{\hat{\mathbf{q}}}$ can be further simplified to

$$S(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}}) = \tilde{d}_{\hat{\mathbf{q}}} - 2\tilde{d}_{\hat{\mathbf{p}}\hat{\mathbf{q}}}, \tag{6}$$

whose value is the smaller the better. The amount of arithmetic computation for evaluating $S(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$ is only $2m+1$ multiplications plus one subtraction for each color channel in the worst case, while n^2 subtractions and n^2 multiplications are needed for the evaluation of the original $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$ in all cases. The worst case happens when all nonzero coefficients in $\Psi_{\hat{\mathbf{p}}}$ have a corresponding nonzero coefficient in $\Psi_{\hat{\mathbf{q}}}$. In general, there are fewer overlapped nonzero coefficients in them. Moreover, the values of $\tilde{d}_{\hat{\mathbf{q}}}$ for all exemplar image blocks can be pre-computed and stored so that during the scoring process, only $\tilde{d}_{\hat{\mathbf{p}}\hat{\mathbf{q}}}$ has to be computed.

To efficiently evaluate $\tilde{d}_{\hat{\mathbf{p}}\hat{\mathbf{q}}}$, we should have a fast method to check if neither $\hat{\mathbf{q}}_{i,j}^{t}$ nor $\hat{\mathbf{p}}_{i,j}^{t}$ are zero. We employ the *search-array* data structure, which was used in [19] for image query. Arrays are constructed in the initialization step for a color channel. For example, let D^t denote the search-array for the color channel L, each element $D^t[i,j]$ of the array contains a list pointing to all images having a nonzero DCT coefficients at (i,j) in the t color channel. $D^t[i,j][k]$ represents the kth image stored at the element $D^t[i,j]$, whose truncated DCT coefficient at (i,j) is nonzero. $D^t[i,j][k].ID$ and $D^t[i,j][k].\alpha$ are the global index of the exemplar image block and its nonzero DCT coefficient at (i,j) respectively.

These search-arrays can be constructed in the initialization step before the main routine of exemplar-based image completion. First of all, all exemplar image blocks in the known region, $I \setminus Q$, are decomposed into the frequency domain by DCT. The DCT coefficients are then truncated by only retaining the m significant ones, and are filled into the search-arrays. Another array, B, named as *base-score array*, is also established during the initialization. $B[k]$ stores the value of $\tilde{d}_{\hat{\mathbf{q}}}$ of the kth exemplar image block $\Psi_{\hat{\mathbf{p}}}$.

By the base-score array and the search arrays, the matching scores of all exemplar images, $\Psi_{\mathbf{q}_k}$ $(k = 0, \cdots, \hat{m}-1)$, to the query image block, $\Psi_{\hat{\mathbf{p}}}$, can be updated by scanning the elements, $D^t[i^*, j^*]$ ($\exists \hat{\mathbf{p}}^t_{i^*,j^*} \neq 0$), of search arrays. Details can be found in **Algorithm 1** *FindBestMatch* (with $k_{\min}$ being the ID of the best matched exemplar).

Algorithm 1 FindBestMatch

```
1: Initialize $scores[k] = B[k]$ for all $k$;
2: for each color channel $t$ do
3:     for each nonzero DCT coefficient $\hat{\mathbf{p}}^t_{i^*,j^*}$ do
4:         for each element $e$ of $D^t[i^*, j^*]$ do
5:             $k = D^t[i^*, j^*][e].ID$;
6:             $scores[k] -= 2(D^t[i^*, j^*][e].\alpha * \hat{\mathbf{p}}^t_{i^*,j^*})$;
7:         end for
8:     end for
9: end for
10: Find $k_{\min} = \arg\min_k scores[k]$;
```

After scoring the exemplar image blocks, we select 0.1% of the total exemplars with smallest scores as candidate exemplars. Among them, the one gives smallest SSD value to $\Psi_{\mathbf{p}}$ on the known pixels is selected as the best matched image block $\Psi_{\hat{\mathbf{q}}}$.

B. Parallel Implementation on GPU

The algorithm presented above maps well on modern graphics hardware, which provides highly parallel computational power. We conducted the CUDA library from nVidia [27] for our parallel implementation. In **Algorithm 1** *FindBestMatch*, parallel implementation of the first step and the most inner loop is trivial when using the technique of *General-Purpose computation on Graphics Processing Units* (GPGPU). The last step in **Algorithm 1** *FindBestMatch* is a standard reduction problem in GPGPU, and it can be computed highly in parallel as well. The basic idea of parallel reduction is to build a balanced binary tree on the input data and sweep it to the root to compute the minimum (or maximum). Details can be found in [28]

The initialization step can also be parallelized to run on GPU. For the exemplar image blocks, we first parallel compute and truncate the DCT coefficients of all image blocks. Then, the entry of the base-score array, B, is also generated in parallel. Afterwards, for each nonzero coefficient $\hat{\mathbf{q}}^t_{i^*,j^*}$ of the exemplar $\Psi_{\hat{\mathbf{q}}_k}$, we let

$$D^t[i^*, j^*][k].ID = k$$

and

$$D^t[i^*, j^*][k].\alpha = \hat{\mathbf{q}}^t_{i^*,j^*}.$$

This parallel assignment is also trivial to implement. Lastly, the empty entries at each element $D^t[i^*, j^*]$ of the search-array are removed by the parallel compaction operation.

Basically, such parallel compaction has three steps (ref. [28]):

1) Initialization: We generate a temporary vector where the elements whose values of ID in the list are nonzero are set to *one* and the other element are set to *zero*.
2) Scanning: Scanning is conducted on the temporary vector by building a balanced binary tree on the vector and sweep it to and from root to compute the prefix sum. The result of scanning then contains the destination address for that element in the output vector.
3) Scattering: The input elements are scattered to the output vector using the address generated by the scan.

All these operations have been recently integrated as primitive elements in the CUDPP library [29], which are easy to use in our image completion program.

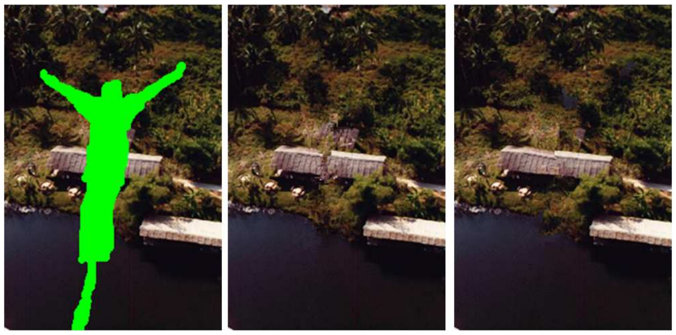


Fig. 4. An example of removing a bungee jumping man: (left) the input image (206 × 308 pixels), (middle) the result of Criminisi's method (in 12.5 sec.), and (right) our result (in 0.9 sec.).

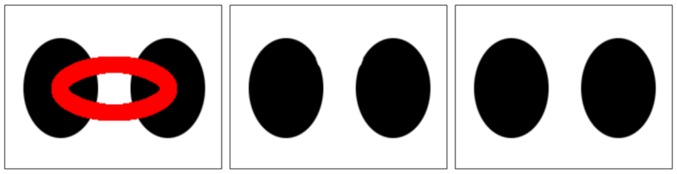
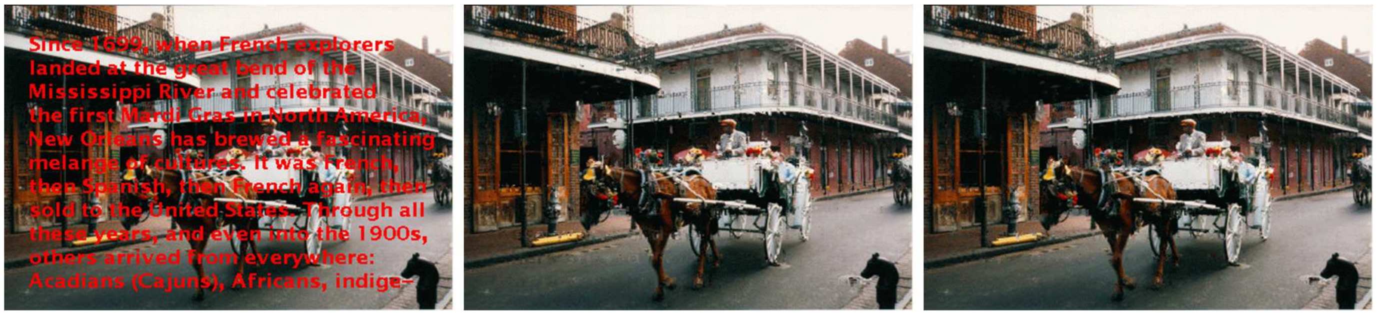


Fig. 6. An example of damaged ellipses: (left) the input image (200 × 150 pixels), (middle) the result of Criminisi's method (in 10.36 sec.), and (right) our result (in 0.67 sec.).

IV. Results and Discussion

We have implemented the proposed algorithm and the one suggested by Criminisi et al. in [2] using Visual C++, and tested them on various examples on a PC with Intel Core 2 Quad CPU Q6600 2.4GHz CPU equipped with a consumer level graphics card – GeForce GTX295. Basically, the proposed fast query method can recover photographs more efficiently than the original exemplar-based approach [2]. All the computations are conducted in the CIE Lab color model. When implementing the original algorithm in [2], we adopt the ANN library [7] to find the best matched patch in each step. Although our method's quality in principle should be worse than that of the original method as we use fewer coefficients, our method is robust and shows better results in some cases. It is because selecting the most significant coefficients in the frequency domain has a denoising effect [30]. Specifically, by the matching method presented in [2], the selected patches to fill may be highly incompatible between the known and the unknown pixels. In our approach, since m-dominant DCT coefficients are used and the unknown pixels are filled smoothly, our matching results are much more compatible and also reflect global visual effect in terms of block instead of pixel.



Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indige-

Fig. 5. An example of removing the unwanted words: (left) the input image with annoying words (438 × 279 pixels), (middle) the result of Criminisi's method (in 62.21 sec.), and (right) our result (in 7.4 sec.).

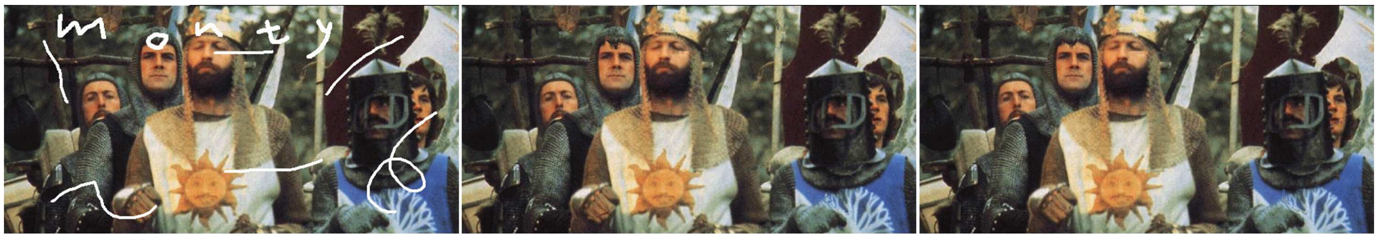


Fig. 7. An example of removing the unwanted drawing: (left) the input image with annoying drawing (628 × 316 pixels), (middle) the result of Criminisi's method (in 47.14 sec.), and (right) our result (in 3.68 sec.).

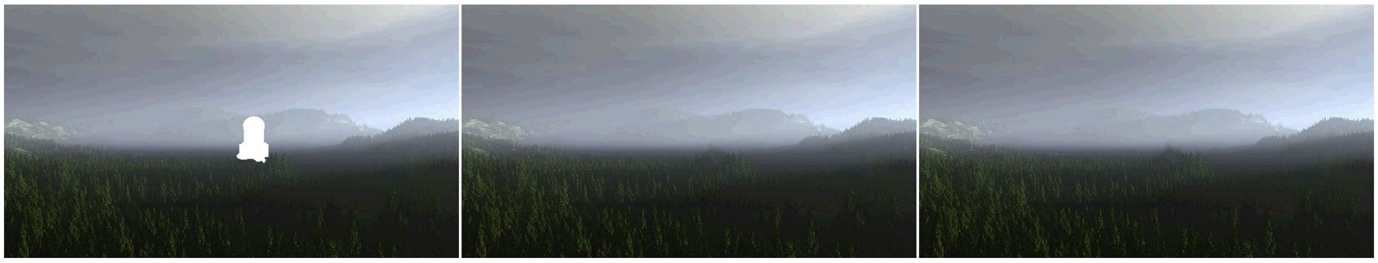


Fig. 8. An example of removing the unwanted building: (left) the input image (700 × 386 pixels), (middle) the result of Criminisi's method (in 76.36 sec.), and (right) our result (in 1.64 sec.).

The patch size of exemplars is critical to the final completion result. The bigger size results in less similar patches, and the smaller size leads to more repeated patterns in filled unknown regions. In order to balance the quality and efficiency, after testing with different examples, we found that the patch size of 9×9 is suitable for most cases. This is also the patch size selected in [2]. For some special cases, like Fig.5, which have small details, a patch size of 5×5 is chosen. Meanwhile, users can also adjust the patch size to fit their needs.

Our first example shown in Fig.1 can be completed in 11.56 seconds by our method but takes about 133 seconds when using our implementation of [2]. More examples are shown in Fig.4-10. It is easy to find that our method can generate similar results as [2] but in a much shorter time. The last example (in Fig.11) is employed to demonstrate that the result of image completion can be improved by specifying the region of searching for candidate exemplars to fill the removed region. Both the quality and speed can be improved by such a simple interactivity.

To verify the efficiency of our method and the quality of resultant images, we compared our results to the state-of-the-art approaches. Our first trial is to employ TSVQ in the patch query. Examples generated by this way are shown in Fig.12, where both balanced and unbalanced binary trees in TSVQ are tested. The speed of using TSVQ is faster than [2] but still slower than ours. The results generated are not as good as ours because that the resultant codeword may not be the optimal one since only part of the tree is traversed for the query.

Considering the algorithm presented in [16], it has reported the results on two common examples as ours – the one shown in Figs.1 and 4. The qualities of results from theirs and ours are similar; however, their method takes 35 seconds (on the baby image - Fig.1) and 8 seconds (on the bungee example - Fig.4). Ours needs only 11.56 and 0.9 seconds respectively.

Lastly, the recently presented random Patch-Match approach [17] is conducted to test the examples shown in this paper without user specified guidance (i.e., the strokes given in the demos of [17]). The image completion in [17] is based on the method of Wexler et al. in [31]. Their method with the help of random Patch-Match [17] can generate results in a very fast speed, and the results in general are acceptable if there is no significant structural edges. Nevertheless, when applying such completion to the images with structural edges, e.g., the bungee example and the damaged ellipses example, the quality

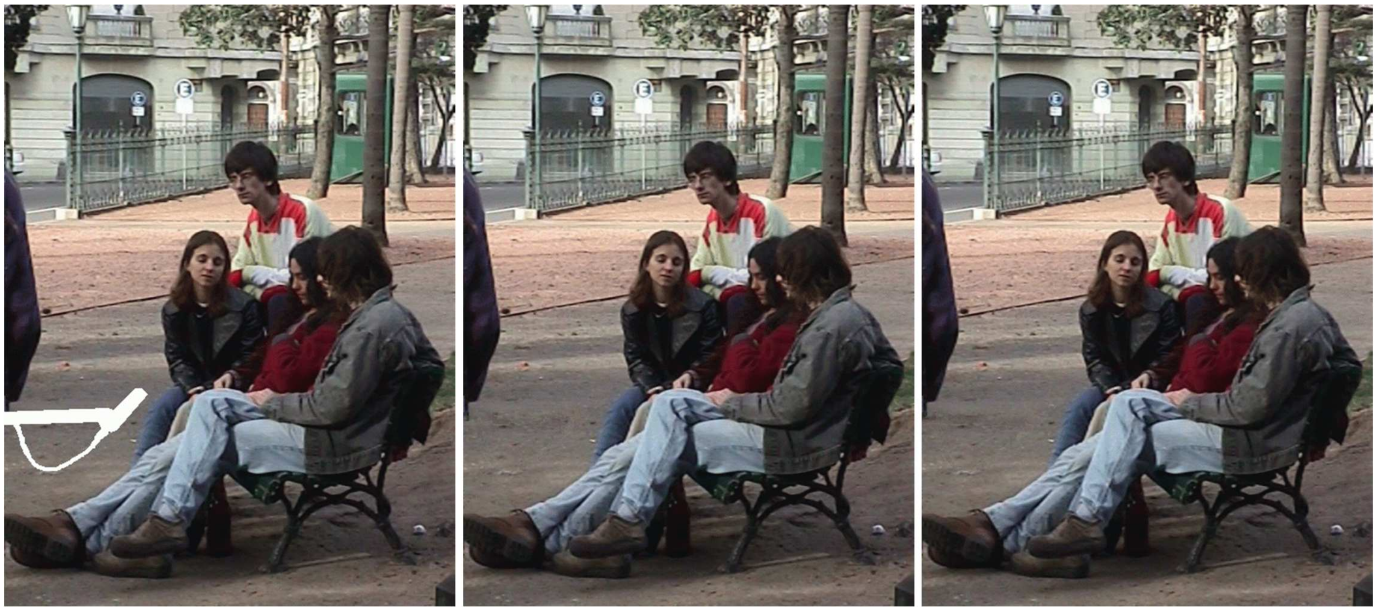


Fig. 9. An example of removing the unwanted microphone: (left) the input image (480 × 640 pixels), (middle) the result of Criminisi's method (in 67.98 sec.), and (right) our result (in 2.48 s.).

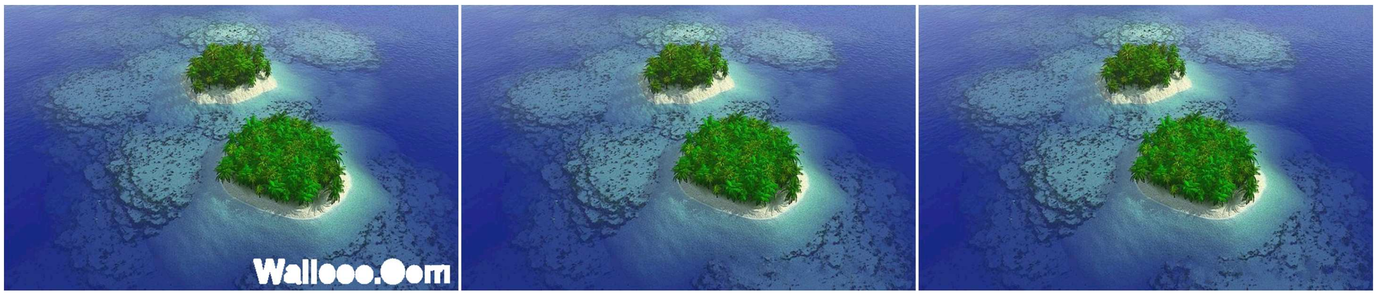


Fig. 10. An example of removing the unwanted words: (left) the input image with annoying words (700 × 438 pixels), (middle) the result of Criminisi's method (in 127.25 sec.), and (right) our result (in 4.18 sec.).

of their results is poorer than ours (see Fig.13). Another similar example is given in Fig.14. Although applying more iterations can somewhat improve the result, the computation does not lead to the straight edge as our approach (rightmost of Fig.14). The image completion results using the random Patch-Match based approach [17] on other two examples (given previously in Fig.1 and 11) are shown in Fig.15.

V. Conclusion

We developed a fast query method under the framework of an exemplar-based image completion in this paper. The exemplars are decomposed into the frequency coefficients and among them a few significant ones are selected to evaluate the matching score. Moreover, a local gradient-based filling algorithm was developed to fill the unknown pixels in a query image block, which improves the accuracy of finding the best matched exemplar. Our new algorithm using the search-array to score the candidate exemplars maps well on the modern graphics hardware with GPU. Experimental tests proved the effectiveness and efficiency of our approach in various photographs.

Acknowledgment

This work was partially supported by the Hong Kong RGC/GRF Grant (CUHK/417508). The author would also like to thank Ms. Siu Ping Mok for proof reading the manuscript.

Appendix A

Lemma If the basic functions of transformation are orthonormal, squared L^2 error of the transformed coefficient differences in two image blocks $\Psi_{\mathbf{p}}$ and $\Psi_{\mathbf{q}}$ is the same as the squared L^2-norm of differences between two images on pixel values.

Proof. Without loss of generality, we use $p_{i,j}$ and $q_{i,j}$ to represent the pixel values in two images, and $F^p_{u,v}$ and $F^q_{u,v}$ to represent their transformed coefficients. They can be expressed as a weighted sum of basis functions

$$F^p_{u,v} = \Sigma_{i=0}^{n-1}\Sigma_{j=0}^{n-1} p_{i,j}\phi_{i,j}(u,v) = \Sigma_{k=0}^{n^2-1} p_k\phi_k(u,v), \quad (7)$$

$$F^q_{u,v} = \Sigma_{i=0}^{n-1}\Sigma_{j=0}^{n-1} q_{i,j}\phi_{i,j}(u,v) = \Sigma_{k=0}^{n^2-1} q_k\phi_k(u,v). \quad (8)$$

The square of L^2 on $F^p_{u,v}$ and $F^q_{u,v}$ can be represented as the inner product of two vectors formed by the components

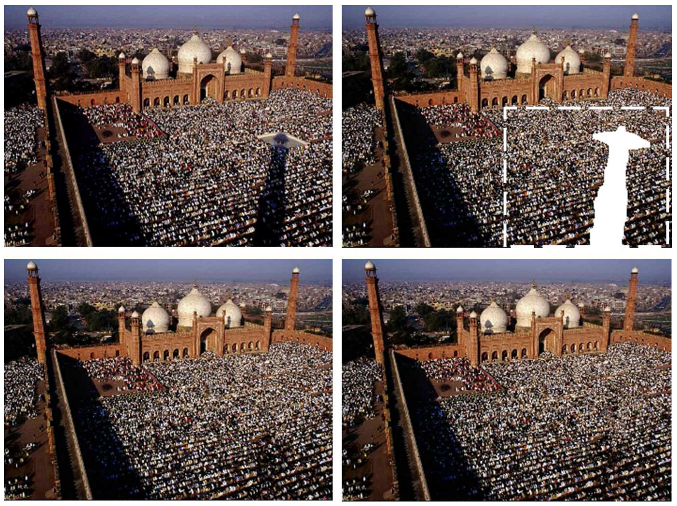
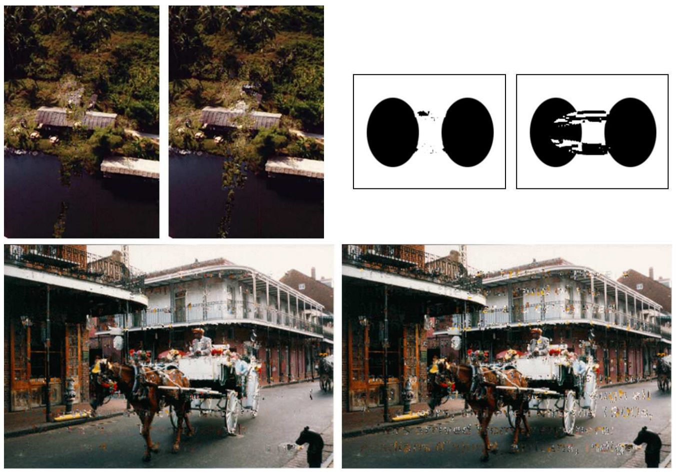


Fig. 11. An example of removing unwanted shadow: (top-left) the original image (350×257 pixels), (top-right) the unwanted shadow has been blanked out, (bottom-left) the result of Criminisi's method (in 2.14 sec. by searching exemplars in the region circled by dashed lines) and (bottom-right) our result of image completion (in 0.41 sec.).



Examples	Unbalanced TSVQ	Balanced TSVQ
Bungee	5.68 sec.	4.97 sec.
Ellipse	1.35 sec.	0.93 sec.
Carriage	18.3 sec.	16.9 sec.

Fig. 12. Results by replacing the exemplar searching step with the TSVQ technique [8]. The completion results using both the unbalanced (left) and the balanced (right) trees in TSVQ are shown, where the unbalanced TSVQ gives better results but takes longer time. The computational statistics in time are listed in the table.

$\{F^p_{u,v}\}$ and $\{F^q_{u,v}\}$, where $\{\cdots\}$ means to span the indices $u, v = 0, \cdots, n-1$.

$$L^2(F^p_{u,v}, F^q_{u,v}) = \Sigma_{u=0}^{n-1}\Sigma_{v=0}^{n-1}(F^p_{u,v} - F^q_{u,v})^2$$
$$= \|\{F^p_{u,v}\} - \{F^q_{u,v}\}\|^2$$
$$=< \{F^p_{u,v}\} - \{F^q_{u,v}\}, \{F^p_{u,v}\} - \{F^q_{u,v}\} >$$
$$=< \{\Sigma_{k=0}^{n^2-1}(p_k - q_k)\phi_k(u,v)\}, \{\Sigma_{l=0}^{n^2-1}(p_l - q_l)\phi_l(u,v)\} >$$
$$= \Sigma_{k=0}^{n^2-1}\Sigma_{l=0}^{n^2-1}(p_k - q_k)(p_l - q_l) < \phi_k(u,v), \phi_l(u,v) >$$

As the basis functions $\phi_k(u,v)$ are orthonormal (i.e., $\phi_k(u,v)\phi_l(u,v) = 1$ only when $k = l$, and $\phi_k(u,v)\phi_l(u,v) =$

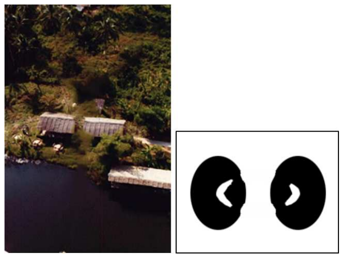


Fig. 13. Results by using the random Patch-Match [17]: (left) the bungee jumping example (in 0.2 sec.), and (right) the damaged ellipses shown in Fig.6 (in 0.12 sec.). As the Patch-Match is a randomized approach, we try several times and show the best result here.

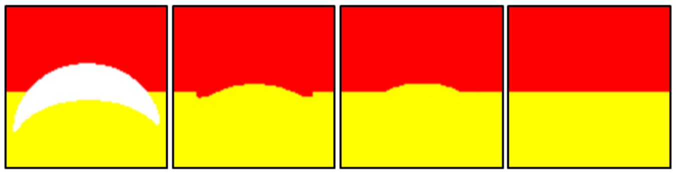


Fig. 14. Results of completing a damaged straight edge: (leftmost) the white region is to be filled (middle-left) the result by random Patch-Match with 5 iterations (in 0.09 sec.), (middle-right) applying more iteration steps (20 iterations in total here) can give better result by Patch-Match, and (rightmost) our method can reconstruct the straight edge (in 0.2 sec.).

1 for any other $k \neq l$), we have

$$L^2(F^p_{u,v}, F^q_{u,v}) = \Sigma_{k=0}^{n^2-1}(p_k - q_k)^2. \quad (9)$$

The right-side of the above equation is the squared L^2-norm of differences between two images $\Psi_{\mathbf{p}}$ and $\Psi_{\mathbf{q}}$ on pixel values.

References

[1] T. Georgiev, "Photoshop healing brush: a tool for seamless cloning," in *Workshop on Applications of Computer Vision (ECCV 2004)*, 2004, pp. 1–8.

[2] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, pp. 1200–1212, September 2004.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *ACM SIGGRAPH 2000*. ACM, 2000, pp. 417–424.

[4] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *IEEE CVPR 2001*, vol. I. IEEE, 2001, pp. 355–362.

[5] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 303–312, 2003.

[6] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," in *IEEE ICCV 2003*. IEEE, 2003, pp. 305–312.

[7] D. Mount and S. Arya, ANN*: A Library for Approximate Nearest Neighbor Searching*. http://www.cs.umd.edu/ mount/ANN/, 2006.

[8] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of SIGGRAPH '00*. New York, NY, USA: ACM Press, 2000, pp. 479–488.

[9] P. Harrison, "A non-hierarchical procedure for re-synthesis of complex textures," in *WSCG 2001 Conference proceedings*, 2001, pp. 190–197.

[10] J. Jia and C.-K. Tang, "Image repairing: Robust image synthesis by adaptive nd tensor voting," in *Proceedings of Computer Vision and Pattern Recognition*, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2003, p. 643.

[11] N. Komodakis and G. Tziritas, "Image completion using global optimization," in *IEEE CVPR 2006*, vol. I. IEEE, 2006, pp. 442–452.

[12] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM Transactions on Graphics (SIGGRAPH 2005)*, vol. 24, pp. 861–868, 2005.

[13] J. Haysm and A. Efros, "Scene completion using millions of photographs," *ACM Transactions on Graphics (SIGGRAPH 2007)*, vol. 26, p. Article 4, July 2007.

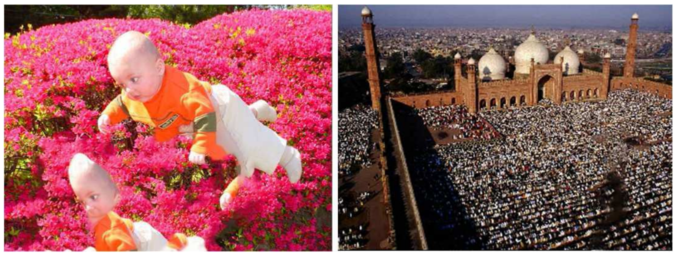


Fig. 15. Results of random Patch-Match based image completion on other two examples. Note that the image shown in the right is completed by the exemplars selected from the region shown in the top-right of Fig.11.

[14] C. Liu, Y. Guo, L. Pan, Q. Peng, and F. Zhang, “Image completion based on views of large displacement,” *The Visual Computer*, vol. 23, pp. 833–841, July 2007.

[15] X. Liu, L. Wan, Y. Qu, T. Wong, S. Lin, C. Leung, and P. Heng, “Intrinsic colorization,” *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, vol. 27, p. Article 152, December 2008.

[16] C.-W. Fang and J.-J. Lien, “Rapid image completion system using multiresolution patch-based directional and nondirectional approaches,” *Image Processing, IEEE Transactions on*, vol. 18, no. 12, pp. 2769 – 2779, Dec. 2009.

[17] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch: a randomized correspondence algorithm for structural image editing,” in *SIGGRAPH ’09: ACM SIGGRAPH 2009 papers*. New York, NY, USA: ACM, 2009, pp. 1–11.

[18] Z.-N. Li and M. Drew, *Fundamentals of Multimedia*. Pearson Prentice Hall, 2004.

[19] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, “Fast multiresolution image querying,” in *SIGGRAPH ’95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1995, pp. 277–286.

[20] S. Lefebvre and H. Hoppe, “Parallel controllable texture synthesis,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 777–786, 2005.

[21] J. Shlens, “A tutorial on principal component analysis,” December 2005.

[22] S. Kumar, M. Biswas, S. J. Belongie, and T. Q. Nguyen, “Spatio-temporal texture synthesis and image inpainting for video applications,” *Image Processing, 2005. ICIP 2005*, vol. 2, pp. II– 85–88, 2005.

[23] E. J. Stollnitz, T. D. Derose, and D. H. Salesin, *Wavelets for computer graphics: theory and applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

[24] B. Bede, H. Nobuhara, and E. D. Schwab, “Multichannel image decomposition by using pseudo-linear haar wavelets,” *Image Processing, 2007. ICIP 2007*, vol. 6, pp. VI – 17 – VI – 20, 2007.

[25] B. Allen, B. Curless, and Z. Popović, “The space of human body shapes: reconstruction and parameterization from range scans,” in *SIGGRAPH ’03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM Press, 2003, pp. 587–594. [Online]. Available: http://dx.doi.org/10.1145/1201775.882311

[26] C.-T. Hsu and J.-L. Wu, “Hidden digital watermarks in images,” *IEEE Transactions on Image Processing*, vol. 8, pp. 58–68, January 1999.

[27] C. Nvidia, “NVIDIA CUDA programming guide,” *http://developr.nvidia.com/cuda*, no. version 2.1, 2008.

[28] M. Harris, S. Sengupta, and J. D. Owens, “Parallel prefix sum (scan) with cuda,” in *GPU Gems 3*, H. Nguyen, Ed. Addison Wesley, Aug 2007.

[29] GPGPU.org, “CUDPP: CUDA data parallel primitives library,” *http://gpgpu.org/developer/cudpp*, no. version 1.0, 2008.

[30] A. Foi, V. Katkovnik, and K. Egiazarian, “Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images,” *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395–1411, May 2007. [Online]. Available: http://dx.doi.org/10.1109/TIP.2007.891788

[31] Y. Wexler, E. Shechtman, and M. Irani, “Space-time completion of video,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, 2007.

Tsz-Ho Kwok received his B.Eng. degree in Automation Computer-Aided Engineering from The Chinese University of Hong Kong (2009). He is currently a PhD student in Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong. His research interests include Image Processing, Geometric and Solid modeling, Visual Computing.

Hoi Sheung received the MPhil degree from the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, in 2009. He is now a PhD candidate at The Chinese University of Hong Kong. His current research interests are reverse engineering, 3d scanning and parallel programming.

Charlie C. L. Wang is currently an Associate Professor at the Department of Mechanical and Automation Engineering, the Chinese University of Hong Kong, where he began his academic career in 2003. He gained his B.Eng. (1998) in Mechatronics Engineering from Huazhong University of Science and Technology, M.Phil. (2000) and Ph.D. (2002) in Mechanical Engineering from the Hong Kong University of Science and Technology. He is a member of IEEE and ASME, and an executive committee member of Technical Committee on Computer-Aided Product and Process Development (CAPPD) of ASME. Dr. Wang has received a few awards including the ASME CIE Young Engineer Award (2009), the CUHK Young Researcher Award (2009), the CUHK Vice-Chancellor’s Exemplary Teaching Award (2008), and the Best Paper Awards of ASME CIE Conferences (in 2008 and 2001). His current research interests include geometric modeling in computer-aided design and manufacturing, biomedical engineering and computer graphics, as well as computational physics in virtual reality.