Constrained Texture Synthesis by Scalable Sub-patch Algorithm

Jhing-Fa Wang, Han-Jen Hsu, Hong-Ming Wang
Multimedia & Communication IC Design Lab
Department of Electrical Engineering, National Cheng Kung University
E-mail: wangjf@csie.ncku.edu.tw, hjhsu@icwang.ee.ncku.edu.tw

Abstract

In this paper, we propose a novel scalable sub-patch algorithm for texture synthesis. Texture synthesis is an important research topic in computer graphics and image processing. We present an efficient algorithm which can past a patch each time without boundary processing technique. Especially, the proposed sub-patch algorithm can be used in constrained texture synthesis to fill the large lost region. On the other hand, the searching neighborhood pixels can be changed due to different patch size in image reconstruction. The comparison of previous algorithms is also provided in this paper. We present a first algorithm used in constrained texture synthesis by regular synthesizing order.

Keywords: Texture synthesis, Sub-patch, Scalable

1. Introduction

Vision perception is one of most important instinct. People are sensitive to the texture of image and video. Texture synthesis can generate similar texture from small input texture. Many texture synthesis algorithms have been presented in previous research. The classification of the algorithms consists of pixel-based, patch-based and feature matching. Pixel-based algorithm synthesizes a pixel each time and needs much computation time. Li-Yi Wei proposed a texture synthesis by fixed-neighborhood searching [1]. The algorithm can be accelerated by tree-structured vector quantization. About patch-based algorithm, Efros proposed a patch-based by image quilting [2]. The overlap part of patch boundary can be decided by minimal error cut calculation. Liang presented a real-time texture synthesis by patch-based sampling [3]. The efficiency of patch-based algorithm is better than pixel-based one. Yu-Hen Hu proposed a block-based method based on the method of Efros which can be used in constrained texture synthesis for image post processing [4]. Some algorithms generate new images by matching the feature in the example texture. Simoncelli generate textures by matching the joint statistics of the image pyramids [5].

In wireless image transmission, the image bitstream may be lost by fading channels. We can use texture synthesis to reconstruct the damaged image. Rane and Bertalmio significantly presented a algorithm combined with texture synthesis and image inpainting which can be used in wireless image reconstruction [6][7]. The searching shape is also block-based in the part of texture synthesis.

In order to decrease the redundant neighborhood pixels in retouching small lost region, we present a scalable sub-patch algorithm. The algorithm can change the neighborhood pixels and patch size. In each step, we past one pixel or one line. In constrained texture synthesis, the algorithm can synthesize texture without boundary artifact.

2. Sub-patch algorithm for constrained texture synthesis

This section will introduce the proposed sub-patch algorithm for constrained texture synthesis. The main feature of the algorithm is pasting the patch by searching the most similar texture. The proposed algorithm is faster than pixel-based algorithm because of pasting a patch each time. After pasting a patch, the algorithm needs not to fix the boundary problem. In order to maintain the advantages of different texture synthesis algorithms, we use an inverse-U shape to capture the probability model of texture. The procedure of the algorithm is shown in Figure 1.

2.1. Parameter definition

For example, we want to synthesize the hole in the image as the original texture. The gray part of Figure 1
is $T_{\text{hole}}$. The dark gray part is the current patch $P_{\text{cur}}$. We take a sample of input texture $T$ to find the best matching neighborhood around $T_{\text{hole}}$. The neighborhood shape is inverse-U to measure the similarity. In each step, we past a best matching patch $P_{\text{can}}$ of $T$ into the synthesized texture $P_{\text{cur}}$ of $T_{\text{hole}}$.

The similarity of $N(P_{\text{cur}})$ and $N(P_{\text{can}})$ is determined by the L2 form which is the sum of squared differences in the neighborhood.

$$D(N_1, N_2) = \sum \left( (R_i(p) - R_j(p))^2 + (G_i(p) - G_j(p))^2 + (B_i(p) - B_j(p))^2 \right)$$

As above, R, G, B are pixels value in red, green and blue channels respectively. The searching neighborhood pixels of the proposed algorithm is scalable because $W_{\text{patch}}$ can be changed in different condition. Table 1 is the list of parameter.

![Figure 1. Procedure of the algorithm](image)

**Table 1. List of parameter**

<table>
<thead>
<tr>
<th>parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{in}}$</td>
<td>2D input texture image</td>
</tr>
<tr>
<td>$T$</td>
<td>2D input texture image without hole</td>
</tr>
<tr>
<td>$T_{\text{hole}}$</td>
<td>Hole of the input texture image</td>
</tr>
<tr>
<td>$P_{\text{cur}}$</td>
<td>Current patch</td>
</tr>
<tr>
<td>$P_{\text{can}}$</td>
<td>Candidate patch</td>
</tr>
<tr>
<td>$N(P_{\text{cur}})$</td>
<td>Neighborhood of current patch</td>
</tr>
<tr>
<td>$N(P_{\text{can}})$</td>
<td>Neighborhood of candidate patch</td>
</tr>
<tr>
<td>$W_{\text{patch}}$</td>
<td>Width of patch</td>
</tr>
</tbody>
</table>

In constrained texture synthesis, the searching neighborhood and synthesizing order will be discussed. We summarize the algorithm in the following pseudocode.

```
function $T_{\text{in}} \leftarrow \text{Sub-synthesis}(T, W_{\text{patch}})$
    For all $P_i \in T_{\text{hole}}$
        Neighborhood($P_i \in T_{\text{hole}}$);
        For all $P_j \in T$
            Neighborhood($P_j \in T$);
            $\text{FindBestSubpatch}(N(P_{\text{cur}}), N(P_{\text{can}}))$;
            $P_{\text{can}} \leftarrow P_{\text{can}}$;
        return $T_{\text{in}}$;

function $N\leftarrow \text{Neighborhood}(P_i)$
    $N(P_i) = \{ \text{Pixel}_{i-1,j}, \text{Pixel}_{i-1,j+1}, \text{Pixel}_{i,j-1}, \text{Pixel}_{i,j+1}, \text{Pixel}_{i+1,j} \}$;
    return $N$;

function $F\leftarrow \text{FindBestSubpatch}(N_1, N_2)$
    $N_{\text{best}} \leftarrow \text{null}$;
    loop all pixels in $N_i$
        if $L2(N_1, N_2) < L2(N_{\text{best}}, N_2)$
        $N_{\text{best}} \leftarrow N_1$;
    return $F$;

* $i$ means the row of the patch and pixels ** $j$ means the column of the patch and pixels

2.2. Scalable searching neighborhood

The quality of the synthesized result is dependent on the size of neighborhood. When we use the algorithm in hole filling, the scalable $W_{\text{patch}}$ can be changed by different hole size. After pasting a line at the first step, we synthesize the next patch of hole continually. An example of constrained texture synthesis in different conditions is shown in Figure 1. The algorithm also can be used in image reconstruction which is shown in Figure 2. When the lost region is a single line in a row, the inverse-U shape can be adjusted as the line. When the lost region is a single window over several columns, the shape can use the same shape from top to bottom. When $W_{\text{patch}}$ is $W_{L1}$, the best matching patch $P_{\text{can}}$ is synthesized to $P_{\text{cur}}$. The synthesizing step is executed...
sequentially. In addition, when $W_{\text{patch}}$ is $W_{\text{LZ}}$, the best matching patch $P_{\text{cur2}}$ is synthesized to $P_{\text{cur1}}$. The total number of neighborhood pixels is $W_{\text{patch}} + 4$.

In this paper, we present a scalable sub-patch algorithm for constrained texture synthesis. The algorithm also can be used to filling gaps in the image. The neighborhood pixels can be changed by different patch size of the lost region of input texture. Most previous methods synthesize the hole from outside to center in spiral order. The novel regular scanning order is the first algorithm in constrained texture synthesis. Our method also does not need much buffer to store the candidate patch. For this reason, the proposed algorithm is more suitable for VLSI implementation. The line-based searching pixels also can be reused in computing the similarity.

![Figure 2. Sub-patch algorithm for image reconstruction](image)

2.3 Synthesizing order

Many constrained texture synthesis algorithm adopt spiral order to synthesize the lost block of texture. In our algorithm, we use raster scanning order from top to bottom. In Figure 2, the synthesizing order includes vertical direction and horizontal direction. In multiple-texture image, different synthesizing order can increase the flexibility of constrained texture synthesis.

3. Experimental results

In order to verify the algorithm, we use MIT VisTex texture set [8] to test the proposed algorithm. The algorithm is implemented by MATLAB. The computation time is about several minutes in Pentium 4, 1.8G MHz. Example of sub-patch algorithm for image reconstruction is shown in Figure 3. We use two directions to fill the lost line of input texture. The algorithm also works in synthesizing the large hole region. Figure 4 and 5 shows the example of sub-patch algorithm for constrained texture synthesis. The size of the input texture is 192 $\times$ 192 and 256 $\times$ 256. The hole size is 96 $\times$ 96 and 128 $\times$ 128. The comparison of the algorithms is shown in Table 2.

4. Conclusion

In this paper, we present a scalable sub-patch algorithm for constrained texture synthesis. The algorithm also can be used to filling gaps in the image. The neighborhood pixels can be changed by different patch size of the lost region of input texture. Most previous methods synthesize the hole from outside to center in spiral order. The novel regular scanning order is the first algorithm in constrained texture synthesis. Our method also does not need much buffer to store the candidate patch. For this reason, the proposed algorithm is more suitable for VLSI implementation. The line-based searching pixels also can be reused in computing the similarity.

Table 2. Comparison of different algorithms

<table>
<thead>
<tr>
<th>Method</th>
<th>Search Pixels</th>
<th>Output pixels</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wei</td>
<td>pixel</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Liang</td>
<td>patch</td>
<td>$4\times(\frac{W_{\text{L}}}{6}) + 4\times\frac{W_{\text{Y}}}{6}$</td>
<td>$W_{\text{B}}^2$</td>
</tr>
<tr>
<td>Hu</td>
<td>patch</td>
<td>$W_{\text{B}}^2 - n$</td>
<td>n</td>
</tr>
<tr>
<td>Ours</td>
<td>Sub-patch</td>
<td>64</td>
<td>1</td>
</tr>
</tbody>
</table>

* $W_{\text{B}}$ is the patch size
** $n$ is the pixels in target region

5. References


Figure 3. Sub-patch algorithm for image reconstruction

Figure 4. Sub-patch algorithm for constrained texture synthesis (vertical, $W_{\text{patch}}=96$)

Figure 5. Sub-patch algorithm for constrained texture synthesis (horizontal, $W_{\text{patch}}=128$)