ImageAdmixture: Putting Together Dissimilar Objects from Groups

Fang-Lue Zhang, Ming-Ming Cheng, *Member*, *IEEE*, Jiaya Jia, *Senior Member*, *IEEE*, and Shi-Min Hu, *Member*, *IEEE*

Abstract—We present a semiautomatic image editing framework dedicated to individual structured object replacement from groups. The major technical difficulty is element separation with irregular spatial distribution, hampering previous texture, and image synthesis methods from easily producing visually compelling results. Our method uses the object-level operations and finds grouped elements based on appearance similarity and curvilinear features. This framework enables a number of image editing applications, including natural image mixing, structure preserving appearance transfer, and texture mixing.

Index Terms—Natural image, structure analysis, texture, image processing

1 Introduction

Image analysis and editing provide an important approach to producing new content in computer graphics. We propose here a dedicated system for users to replace structured objects selected from groups while preserving element integrity and providing visual compatibility. It proffers an unconventional way to create new image content with structured objects, even when they are irregularly arranged. The objects are allowed to be dissimilar in appearance and structure, as shown in Fig. 1.

Image and texture synthesis from multiple exemplars have been explored in previous work. Pixel-level texture approaches make use of similar statistical properties to generate new results [4], [9], [45]. Patch-based texture synthesis [14] is another set of powerful tools to selectively combine patches from different sources [23]. Recently, Risser et al. [35] employed multiscale neighborhood information to extend texture synthesis to structured image hybrids. They are useful for creating new image or texture results based on examples.

These methods are general; but in the context of object replacement within groups, they may not be able to produce visually plausible effects without extensive user interaction. Grouped elements can be rapidly and integrally perceived by the human visual system (HVS) [21], which indicates only considering pixel- or patch-level operations may not be enough to understand the individual structured element properties. Our system is composed of object-level editing algorithms and does *not* assume any particular texture form or near-regular patterns. Our major objective is to deal with natural images containing piled or stacked objects, which are common in real scenes, from the individual perspective.

The naturalness of element replacement depends on the compatibility of input and the result structures. It is measured in our system by the element-separability scores, combining curvilinear features and multiscale appearance similarity. Objects are extracted based on these metrics, with initial center detection followed by a Dilate-Trim algorithm to form complete shapes. Our method also finds suitable candidates for element replacement to ensure the result quality. Our main contribution lies in an object-level manipulation method without regularity assumption and on a new scheme to measure object structure compatibility.

2 Related Work

Image composition is a well-studied problem in computer graphics. Early work of Porter and Duff [33] used an alpha matte to composite transparent objects. Recent advance in alpha matting [41], [42] made it possible to generate natural and visually plausible image composite with global or local operation. For our problem, directly apply alpha matting to replace one object by another can generate visual artifacts when illumination conditions vary. Poisson blending [32] and its variations reduce color mismatching by computation in the gradient domain. Farbman et al. [16] achieved similar composition results efficiently. Research efforts have also been made to select candidates for composition from image database [25] or internet images [10], which was applied to create a fantastic artform called Arcimboldo-like collage in [11]. Such methods minimize composition artifacts while keeping the original object shape complete. In our problem, it is essential to seek proper image content to replace items in groups via compatible structures. We also do not simply copy the whole region to the target image for blending. There are procedures to refine boundaries to improve the visual quality of the results when in common situations that the found source and target objects are diverse in structure.

- *F.-L. Zhang, M.-M. Cheng, and S.-M. Hu are with the TNList, Tsinghua University, Beijing 100084, China. E-mail: z.fanglue@gmail.com, cmm@qq.com, shimin@tsinghua.edu.cn.*
- *J. Jia is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Room 1018, Ho Sin Hang Engg. Bldg., Shatin, N.T., Hong Kong. E-mail: leojia@cse.cuhk.edu.hk.*

Manuscript received 27 Aug. 2011; revised 17 Jan. 2012; accepted 17 Jan. 2012; published online 17 Feb. 2012.
Recommended for acceptance by M. Agrawala.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-08-0201.
Digital Object Identifier no. 10.1109/TVCG.2012.68.

Fig. 1. A mixture created by our approach. Cookies are replaced by plums while maintaining a visually plausible structure.

Our work is also related to example-based texture synthesis. Efros and Leung [15] proposed a nonparametric pixel-wise method to synthesize new texture from a user provided example, trying to preserve as much local structures as possible. The efficiency and capability were improved by tree-structured vector quantization [44], global optimization [22], and multiscale synthesis [17], [26]. Patch-based approaches [14], [23] form another important line for texture synthesis with the main idea being to find an optimal seam between adjacent patches. Mask images [31], [47] and feature maps [37], [46] can be used to guide texture synthesis. These methods provide a powerful tool for creating visually similar content from a few examples [43], but none of them can be applied directly to natural images for separating grouped elements, and then replacing objects in a visually plausible way. The main challenge is that the underlying structured element regions may be changed in this process, as demonstrated in Fig. 1.

Synthesis and analysis may also be done at the texel level [34]. Dischler et al. [13] manually segmented texture into particles. Automatic methods [1], [40] extracted texels based on subtree matching in low-level segmentation. They can extract texels that are significantly occluded. It is, however, time-consuming and can fail when segmentation is unreliable for grouped elements. Ijiri et al. [20] showed how to manipulate elements which are distributed near-regularly.

Combining features or visually meaningful content from different samples, i.e., image/texture mixing, allows users to produce results with large variation. In prior work such as image hybrids [35], Risser et al. synthesized image hybrids from exemplars. Other methods [4], [9], [45] generated new texture from multiple examples. The mixed area produced by these methods tends to have similar statistics or visual characteristics to the source. In RepFinder [12], although element replacement can be conceptually achieved, strict shape similarity has to be enforced and there is no consideration of object consistency. In contrast, our work places elements from different groups, where the visual characteristics are allowed to be greatly different. Recently, Ma et al. [30] mixed objects by modeling the combination of individual elements and their distributions. However, this method cannot solve our problem and requires users to manually produce the source elements.

3 Grouped Elements Analysis

Replacing content within groups of objects is challenging, since our human vision system (HVS) is sensitive to scene integrity, shape compatibility, illumination consistency, etc. Our method first finds separable objects in the target image based on element-separability analysis, which captures the global distribution of grouped elements. An *element-separability map* is then generated, indicating where to find individual elements. We use a Dilate-Trim method to extract element boundaries.

3.1 Multiscale Appearance Similarity

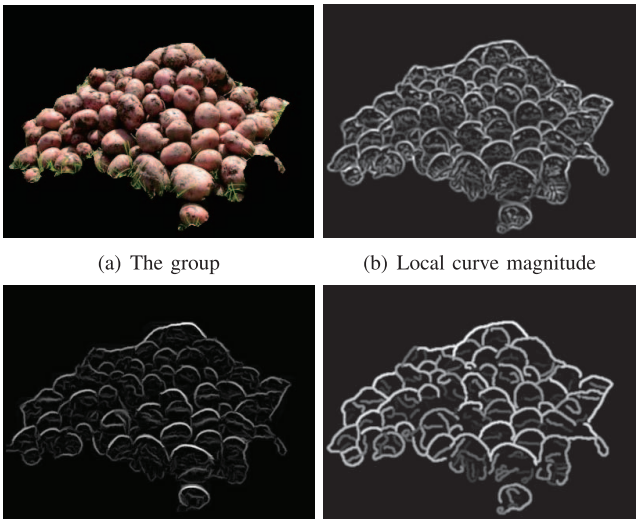
The first stage of our method is to detect inherent appearance similarity in the given target group, which helps analyze element distribution. The necessity of doing this has been observed in other texel analysis work [8], [18], [28], [48].

Our system requires the user to choose a point p_0 to specify a key appearance feature of the objects to be extracted, as shown in Fig. 2a. We use multiscale descriptors [35] to measure how this point (together with its neighbors at different scales) matches other parts of the given image, which eventually lets us know the possible distribution of elements in the group. We build a Gaussian pyramid. The finest level is the original image. Each coarser level is smoothed and down-sampled by a factor of 2. The descriptor vector of pixel p at each level has 75D after stacking all pixels in a 5×5 square neighborhood window in a vector considering three colors. The vector at each level is projected to only 6D by principal component analysis (PCA). If the total number of levels is L, concatenating all 6D vectors in the pyramid yields a $6L$-D final descriptor. Our experiments show that $L = 4$ is generally a good choice. Using a larger L can capture a wider range of visual features with higher computational costs.

We measure the euclidean distances between the descriptor vector for p_0 and those for other pixels, which yield an appearance similarity map S_a with the same size as

(a) Input image (b) Appearance similarity (c) Element-separability (d) Detected elements (e) Replacement result

Fig. 2. Framework: (a) the user selects one point, and our system generates an appearance similarity map (b), informative for finding objects in groups. After curvilinear feature refinement, we obtain the separability map (c). In (d), yellow dots represent elements detected by our method, with their core regions marked in blue. A replacement result is shown in (e).



(a) The group (b) Local curve magnitude

(c) Curve saliency [7] (d) Our result

Fig. 3. Comparison of curvilinear feature detection methods. The local curve magnitude method [39] and the message-passing-based curve saliency map [7] are shown in (b) and (c). Our result is shown in (d).

image I. For each pixel p, its value is denoted as $S_a(p, p_0)$. An example is shown in Fig. 2b. Dark pixels are with high similarity with p_0 with respect to multiscale neighbors.

3.2 Robust Curvilinear Features

Curvilinear features [46], such as edges and ridges, which primarily describe object boundaries in natural images, provide strong evidence for finding similar objects. As shown in Fig. 3b, curve magnitudes in natural images are influenced by object occlusion, illumination variation, local appearance change, etc. The curve detection method [39] depends on local curve magnitudes and may find it difficult to maintain boundary continuity.

Perception study shows that long coherent curves are perceptually salient to HVS [6]. Bhat et al. [7] employed a message passing scheme to find such curves. Albeit effective, it occasionally breaks object boundaries, hindering object extraction and transplantation.

We propose a simple method to define curve saliency according to curve length and average local curve magnitude. Our approach starts by finding curvilinear structures [39], which link confident curve points using orientation matching. This procedure yields local curve magnitude m_p for each pixel p (shown in Fig. 3b), the length l_C of each curve C, and corresponding curve points. Based on these, we define saliency for each curve C as

$$S_c(C) = \mathcal{N}\left(\frac{1}{|C|}\sum_{p\in C} m_p\right) \cdot \mathcal{N}(l_C), \tag{1}$$

where $\frac{1}{|C|}\sum_{p\in C} m_p$ is the average gradient magnitude along C, as $|C|$ is the number of pixels in C. $\mathcal{N}(*)$ is a Gaussian normalization function [2], keeping 99 percent of the values in range $[0, 1]$ after conversion. $S_c(C)$ has two terms, so that a long curve with large saliency is favored. Combining $S_c(C)$ for all curves C, we form a curve saliency map S_c. Each pixel p has a value denoted $S_c(p)$.

As shown in Fig. 3d, our curve saliency map $S_c(I)$ contains more informative structures than those of Bhat et al. [7], and exhibits coherent curvilinear features. In addition, our saliency definition is suitable for user interaction because it only takes a few milliseconds of computation for a typical natural image, while in [7], tens of seconds are needed.

3.3 Separation of Structured Objects

The position of each element in the target group is a potential place to put a new item. Both the appearance similarity and curvilinear features are important cues for the HVS. We estimate an element separability map (ES-map for short) to identify element centers, defined as

$$S(p) = S_c(p) - \omega S_a(p, p_0), \tag{2}$$

where $S_c(p)$ and $S_a(p, p_0)$ are the curve saliency and appearance similarity values for p, respectively. The weight ω is set to 1 in experiments. A smaller ω makes the curvilinear features more readily separate neighboring elements, under the risk of partitioning one element into two.

The simple ES map is vital for understanding the global distribution of all elements. Regions with high-similarity of appearance will present local peaks in the map, while still being outlined by the curvilinear features. With this map, elements close to each other can be quickly separated, as shown in Fig. 2c.

We perform clustering-based segmentation in the ES map, by thresholding the map and taking each connected region as an individual segmented component. The default threshold is set to 0.5. If the threshold is too small, neighboring elements with unclear boundaries could be recognized as one. In the segment map M_s, each region is basically an element's core body. The region centroids can thus be found, highlighted in yellow in Fig. 2d. The region boundaries are, however, coarsely determined, and will be refined in the next step.

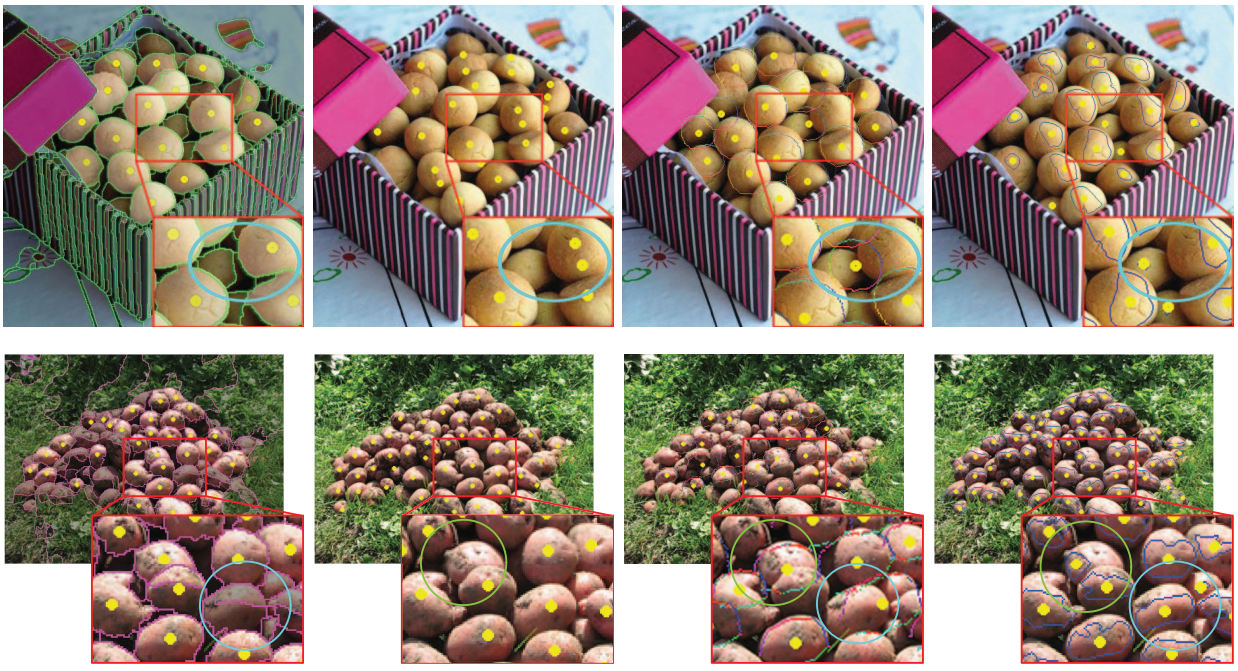
3.3.1 Comparison with Other Methods

2.1D-texels. The element extraction result of 2.1D-texels [1] is shown in Fig. 4a for comparison. This method applies matching in the segmentation tree to achieve unsupervised element extraction. As the elements may not have similar subtrees, missing objects and false detection can be more seriously resulted for this challenging example.

Region of dominance (RoD). Region of Dominance was employed in [27] to extract texels in near-regular textures based on normalized cross correlation (NCC). RoD is defined as the largest circle centered at the candidate peak of NCC. Due to search for dominant peaks, if two elements are very close, separation cannot be achieved, as shown in Fig. 4b. Further, RoD does not provide the boundary information, which is nonetheless critical for our object transplantation in groups.

RepFinder. RepFinder was proposed in [12] to detect approximately repeated elements in natural scenes. A Boundary Band Map (BBM) was employed to extract objects. However, because RepFinder relies on the shape similarity to extract objects, it fails when the grouped elements have different shapes, as shown in Fig. 4c.

Note that our method does not assume that the elements are regular in distribution and in appearance. The element-separability analysis is applicable to examples that are challenging for near-regular texture methods, and



(a) 2.1D texels (b) RoD (c) RepFinder (d) Our result

Fig. 4. Comparison of 2.1D texels [1], Region of Dominance after NCC [18], RepFinder [12] and our method. Yellow dots are the detected element centers.

provides a reliable object separation scheme in general. More results are shown in later sections.

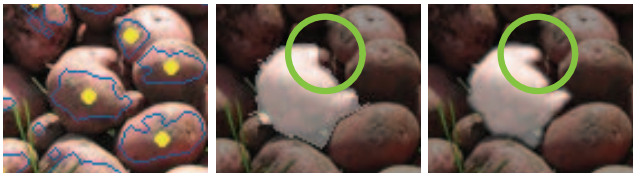
3.3.2 Element Extraction by Dilate-Trim

Given the computed element centers, i.e., yellow dots in Fig. 4, we now refine each element's boundary L by requiring it to have large gradient magnitude and small irregularity. The active contour model provides one way to solve this problem. The summed external energy term of every contour point is

$$E_L = \int_L \frac{S_c(p)}{D(p)S_a(p,p_0)+\epsilon} ds = \sum_{p\in L} \frac{S_c(p)}{D(p)S_a(p,p_0)+\epsilon}, \quad (3)$$

where ϵ is a small value to avoid division by zero and $D(p)$ is the distance from pixel p to the corresponding center point. However, as Fig. 5 shows, because the active contour method trades off between the contour smoothness and external energy distribution, it cannot cover all the details.

We thus use a Dilate-Trim method, sketched in Algorithm 1, to solve the problem progressively. Specifically, it iteratively dilates regions and then trims them until the added pixels are less than an adaptive threshold $t(p) = S(p)/D(p)$. $t(p)$ makes dilation be stopped for pixels with high separation confidence or distant enough from the center. The border gradually expands from the element center, and can evolve differently for various objects, suitable for forming irregular shapes. As Fig. 5 shows, in comparison with the active contours, our method can adaptively and more accurately update object boundaries, fitting our pursuit.



(a) Element (b) Result of snake (c) Our result

Fig. 5. Comparison of the active contour method and our Dilate-Trim. (a) Separable elements with their core regions. (b) Mask from active contour. The external energy is calculated by (3). (c) Our automatically produced result.

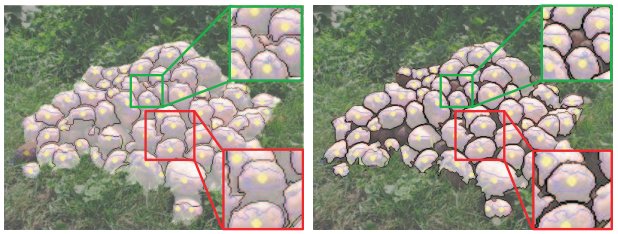
Algorithm 1. Extract an elementary region R by Dilate-Trim

```
Initialize R as one region in M_s in Section 3.3;
Initialize R' = ∅;
while R − R' > δ, where δ is a stopping value do
    R' ← R; R ← Dilate(R);
    for every pixel p ∈ R do
        remove p from R, when S_c(p) < t(p);
    end for
    R ← largest connected region of R;
end while
```

Another popular choice for multilabel segmentation in element groups is by global optimization, such as graph cuts and watershed, taking the element centers as seeds. This scheme cannot handle pixels that do not belong to any of the elements. One example is shown in Fig. 6a. Our method, on the contrary, considers individual elements starting from their centers, and thus can produce better shapes as shown in Fig. 6b. Pixels are allowed to be not in any of the elements.

3.3.3 Choosing Suitable Elements

We perform Dilate-Trim for each element in the target image, exemplified in Fig. 2a, containing a group of objects.



(a) Global optimization (b) Dilate-Trim

Fig. 6. Global segmentation versus dilate-trim. (a) Result of multilabel segmentation. (b) Our result of dilate-trim on each element starting from its center.

Based on the results, our system allows the user to choose the element region to replace. Our method then finds a candidate region from the secondary image with similar curvilinear feature and compatible with the target gradient change.

The similarity between the two images is measured by cross-correlation. We expand the target object boundary by a few pixels, creating a boundary band map [12]. Then we calculate cross correlation values between the target element and all regions in the secondary image on the bands. The two regions that yield the largest value are regarded as most compatible. If there are multiple objects to be processed, we repeat this selection process.

4 Element Replacement

To replace an element R^A in image A by R^B obtained from the secondary image B, object shape needs to be maintained with natural visual appearance change. This goal cannot always be achieved successfully when using gradient-domain image blending. We instead propose a patch searching strategy to retain visually compatible borders. The luminance is then transferred to produce final results.

Poisson blending and tone mapping are useful tools in image composition. The need to keep object appearance in our problem, however, is beyond their capability. First, these methods can drastically change the object color when the source and target images are quite different. Second, they cannot create natural boundaries for each replaced object, as shown in Fig. 1.

4.1 Boundary Replacement

With the candidate region R^B found in the secondary image B, as described in Section 3.3, we initially replace R^A by R^B,

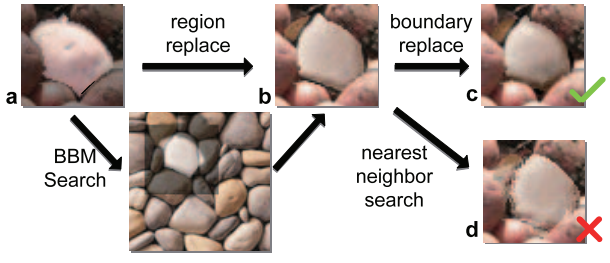


Fig. 7. Element replacement. (a) Region extracted by Dilate-Trim. (b) Initially replacing R^A by R^B. (c) Our final $R^{A'}$ with boundary refinement on (b). (d) Boundary refinement by patch-based nearest neighbor search and replacement, which is visually less pleasing than (c).

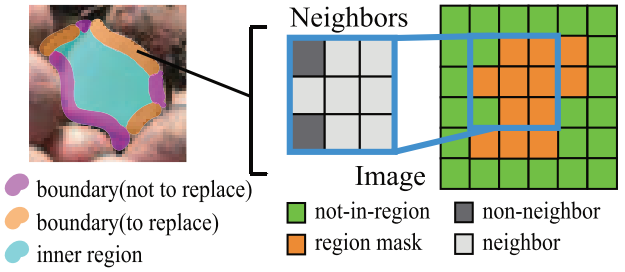


Fig. 8. Three different areas in an object (left). In the boundary band to be refined, the mask only contain pixels inside the object (right).

as shown in Fig. 7b. R^B, by definition, contains the most similar curvilinear feature as R^A near boundary. Note that directly copying R^B still causes visual artifacts. One intuitive way to improve it is by employing the nearest neighbor search, like PatchMatch [5], to refine the boundary of R^B. However, as shown in Fig. 7d, this method does not perform well enough when A and B do not have similar structures.

We alternatively look for large intensity variation near boundaries of R^B and R^A. The updated region, denoted as $R^{A'}$, is produced as

$$R_p^{A'} = \begin{cases} R^B_{\arg\min_q D(p,q)} & p \in \Omega(R^A) \bigcap \left\| \nabla_p^B - \nabla_p^A \right\| > T \\ R_p^B & \text{otherwise,} \end{cases} \quad (4)$$

where $\Omega(R^{A'})$ is the 5-pixel width boundary band of $R^{A'}$, as shown in Fig. 8. q can be any pixel in R^B, and $\|\nabla_p^B - \nabla_q^A\|$ measures difference in gradient. T is a threshold, set to 0.25 by default; our algorithm is insensitive to this value. $D(p,q)$ is the distance between p and q, defined as

$$D(p,q) = \left\| N_q^B - N_p^A \right\| + \alpha \left\| \nabla_q^B - \nabla_p^A \right\|, \quad (5)$$

where N_q^B and N_p^A are masked local windows centered at q and p, respectively. We use $\alpha = 1$ in all our experiments to equally weight the two factors. Pixels p and q have to be close in both appearance and gradient to yield a small $D(p,q)$ along the region's boundary $\Omega(R^{A'})$.

To design N_q^B and N_p^A, square windows [36], [45] used in traditional texture synthesis are not appropriate when irregular boundaries exist. In our method, we only include pixels that are in region R^A for reliable appearance matching, as illustrated in Fig. 8.

So the overall object replacement can be summarized as a two-step process, which first copies pixels from R^B to R^A. Then part of the boundary (shown in orange on the left of Fig. 8a) is adjusted according to (4) to improve appearance compatibility. Fig. 7c shows a result.

In (4), to obtain q w.r.t. $\min_q D(p,q)$, we avoid time-consuming brute-force search in B and only keep candidate patches in R^B that have small mean-color differences to N_p^A. We maintain the top 20 percent candidate patches for detailed comparison in (4).

4.2 Luminance Transfer

Finally, we approximate luminance from the original image, and transfer it to the result. Existing luminance transfer methods [24] mainly consider features and image structure. Since we already have the appearance similarity value $S_a(p, p_0)$ for every pixel indicating the difference between p

Fig. 9. Mixing elements. (a) Target image. (b) ES-map. (c) Detected elements. (d) Secondary image. (e) Mixing result.

and the user selected pixel p_0, we define a luminance score function $L(p)$ for every pixel p:

$$L(p) = S_a(p, p_0) \cdot \left(\bar{I}_{N_p^A} - \bar{I}_{N_{p_0}^A}\right), \tag{6}$$

where N_p^A denotes the masked neighborhood of p in image A, and $\bar{I}_{N_p^A}$ is the average intensity of pixels in N_p^A. $L(p)$ can be either positive or negative, and is useful to change the pixel brightness depending on its sign. The final luminance transfer is performed for each pixel p using

$$P'(p) = P(p) + \lambda L(p), \tag{7}$$

where $P(p)$ and $P'(p)$ denote the pixel values before and after luminance transfer, respectively, and λ controls the level of modification. Large λ yields strong highlight or shadow. We use $\lambda = 0.75$ in all our experiments. Results before and after luminance transfer are shown in Fig. 10.

5 Applications and Results

In our algorithm implementation, the regions to operate on in the input images are segmented by grab-cut [38] if necessary. Then the target objects are separated as described in Section 3. The specified objects are replaced one by one using the method given in Section 4. On a PC with a CPU at 2.5 GHz and 4 GB RAM, the computation time is around 0.2 seconds for element analysis on a 640×480 image, and is about 20 milliseconds to replace each element.

(a) Before luminance transfer (b) Luminance transfer result

Fig. 10. Results before and after luminance transfer.

Fig. 11. Structure preserving texture transfer: (left) target image and novel source texture; (right) texture transfer result.

5.1 Natural Image Examples

In real scenes, it is ubiquitous to see grouped elements with similar appearance. Our method is able to produce high-quality element replacement results, as shown in Fig. 9. Unlike image composition techniques, our approach focuses on mixing visual features into the target area, while keeping the original structures, rather than overwriting them. Our image mixture approach is a complement to current image editing techniques.

5.2 Structure Preserving Appearance Transfer

Our method can also be applied to structure preserving appearance transfer in textured images. We only separate elements in the target image group using the boundary band map [12] to search for most compatible structures in the secondary texture image for replacement. Thereupon, the secondary image is not limited to grouped elements. One example is shown in Fig. 11, where we use a lichen

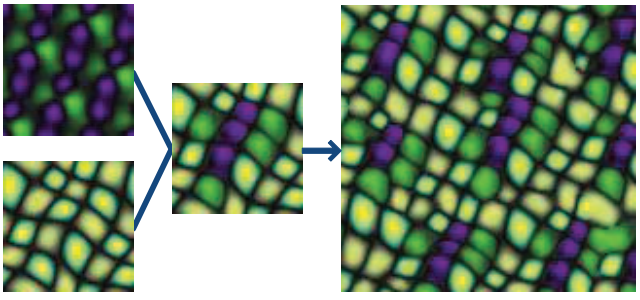


Fig. 12. Texture mixing result. The user inputs two different images, based on which we produce a mixing result. A larger texture can be synthesized from the result.

Fig. 13. Failure cases: (left) grouped objects with complicated 3D structure and layers; (right) object groups with incompatible structure.

Fig. 14. Further results. Each group has three images showing (from left to right) the source image, the secondary image, and our result, respectively.

texture to replace bricks, accomplishing the appearance transfer effect. The major difference between our approach and texture transfer [14], [19], [29] is that we can preserve the appearance and local structure of the original objects that remain in the admixture result. In Fig. 11, lichen appears to be growing on bricks.

5.3 Texture Mixing

To blend texture, prior methods require input samples to have similar appearance or weak structural features [4], [9], [45]. Our approach is an alternative to this end. We first mix the two exemplars to create a new texture sample using the steps described in the paper. Each texton in the two textures is regarded as an atomic element. Then, we perform the texture synthesis approach of Lefebvre and Hoppe [26] to generate a bigger image. One example is shown in Fig. 12, our method spatially blends two different kinds of texels in an element-wise manner, rather than providing a statistically uniform result similar to both inputs. More results of the above applications are shown in Fig. 14.

5.4 Limitations

There are a few difficulties that might influence the final result quality. First, since we do not estimate any depth or layer information, our system cannot separate group objects well when this set of information is needed. We show an example in the left of Fig. 13, in which leaves overlay, destroying the latent integrity. The lack of depth information may also make object replacement not visually very pleasing when the 3D structure significantly varies in the two input images.

Secondly, we require objects in the secondary image to have curvilinear compatible content to replace the original elementary region. As shown in the right of Fig. 13, when we replace the rectangular blocks by hexagonal stones, even considering the most compatible regions cannot preserve the visual integrity. The incomplete rocks inserted into the wall do not retain their correct geometric features. Finally, challenging objects like transparent or translucent glasses, cannot be properly handled in our system. Involving matting and image recognition will be our future work.

6 Conclusion

In this paper, we have proposed a dedicated system to substitute structured elements within groups by others even though they are dissimilar in appearance. Different levels of mixture are allowed in our system, preserving local structures. Our method requires a very small amount of user interaction. The analysis and other operations to separate objects are automatic. The drawback is that we do not consider layers and depth, which may be problem in some cases. But overall our method is powerful enough to handle many examples that are challenging for other approaches.

Acknowledgments

We thank all the reviewers for their helpful comments. This work was supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Numbers 61120106007 and 61103079).

References

[1] N. Ahuja and S. Todorovic, "Extracting Texels in 2.1D Natural Textures," *Proc. IEEE 11th Int'l Conf. Computer Vision (ICCV)*, pp. 1-8, 2007.

[2] S. Aksoy and R. Haralick, "Feature Normalization and Likelihood-Based Similarity Measures for Image Retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563-582, 2001.

[3] W.-C. Lin and Z.-C. Yan, "Attention-Based High Dynamic Range Imaging," *The Visual Computer*, vol. 27, no. 6, pp. 717-727, 2011.

[4] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans Visualization and Computer Graphics (TVCG)*, vol. 7, no. 2, pp. 120-135, Apr.-June 2001.

[5] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Trans Graphics*, vol. 28, pp. 24:1-24:11, 2009.

[6] W. Beaudot and K. Mullen, "How Long Range Is Contour Integration in Human Color Vision?" *Visual Neuroscience,* vol. 20, no. 1, pp. 51-64, 2003.
[7] P. Bhat, C. Zitnick, M. Cohen, and B. Curless, "GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering," *ACM Trans Graphics,* vol. 29, no. 2, pp. 1-14, 2010.
[8] S. Brooks and N. Dodgson, "Self-Similarity Based Texture Editing," *ACM Trans Graphics,* vol. 21, pp. 653-656, 2002.
[9] G. Brown, G. Sapiro, and G. Seroussi, "Texture Mixing Via Universal Simulation," *Proc. Int'l Workshop Texture Analysis and Synthesis,* 2005.
[10] T. Chen, M. Cheng, P. Tan, A. Shamir, and S. Hu, "Sketch2-Photo: Internet Image Montage," *ACM Trans Graphics,* vol. 28, no. 5, pp. 124: 1-10, 2009.
[11] H. Huang, L. Zhang, and H.-C. Zhang, "Arcimboldo-Like Collage Using Internet Images," *ACM Trans Graphics,* vol. 30, no. 6, pp. 155:1-155:8, 2011.
[12] M.-M. Cheng, F.-L. Zhang, N.J. Mitra, X. Huang, and S.-M. Hu, "Repfinder: Finding Approximately Repeated Scene Elements for Image Editing," *ACM Trans Graphics,* vol. 29, no. 4, pp. 83:1-8, 2010.
[13] J.-M. Dischler, K. Maritaud, B. Lvy, and D. Ghazanfarpour, "Texture Particles," *Computer Graphics Forum,* vol. 21, pp. 401-410, 2002.
[14] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis And Transfer," *Proc. ACM SIGGRAPH,* pp. 341-346, 2001.
[15] A.A. Efros and T.K. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Seventh Int'l Conf. Computer Vision (ICCV),* pp. 1033-1038, 1999.
[16] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski, "Coordinates for Instant Image Cloning," *ACM Trans Graphics,* vol. 28, pp. 67:1-9, 2009.
[17] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun, "Multiscale Texture Synthesis," *ACM Trans Graphics,* vol. 27, no. 3, pp. 1-8, 2008.
[18] J. Hays, M. Leordeanu, A. Efros, and Y. Liu, "Discovering Texture Regularity as a Higher-Order Correspondence Problem," *Proc. Ninth European Conf. Computer Vision (ECCV),* pp. 522-535, 2006.
[19] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies," *Proc. ACM SIGGRAPH,* pp. 327-340, 2001.
[20] T. Ijiri, R. Mch, T. Igarashi, and G. Miller, "An Example-Based Procedural System for Element Arrangement," *Computer Graphics Forum,* vol. 27, pp. 429-436, 2008.
[21] B. Julesz, "Textons, The Elements of Texture Perception, and Their Interactions." *Nature,* vol. 290, no. 5802, pp. 91-97, 1981.
[22] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Example-Based Synthesis," *ACM Trans Graphics,* vol. 24, pp. 795-802, 2005.
[23] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans Graphics,* vol. 22, pp. 277-286, 2003.
[24] J.-F. Lalonde, A. Efros, and S. Narasimhan, "Estimating Natural Illumination from a Single Outdoor Image," *Proc. IEEE 12th Int'l Conf. Computer Vision (ICCV),* pp. 183-190, 2009.
[25] J.-F. Lalonde, D. Hoiem, A.A. Efros, C. Rother, J. Winn, and A. Criminisi, "Photo Clip Art," *ACM Trans Graphics,* vol. 26, 2007.
[26] S. Lefebvre and H. Hoppe, "Parallel Controllable Texture Synthesis," *ACM Trans Graphics,* vol. 24, pp. 777-786, 2005.
[27] Y. Liu, R. Collins, and Y. Tsin, "A Computational Model for Periodic Pattern Perception Based on Frieze and Wallpaper Groups," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 26, no. 3, pp. 354-371, Mar. 2004.
[28] Y. Liu, J. Wang, S. Xue, X. Tong, S. Kang, and B. Guo, "Texture Splicing," *Computer Graphics Forum,* vol. 28, no. 7, pp. 1907-1915, 2009.
[29] Y. Liu, W.-C. Lin, and J. Hays, "Near-Regular Texture Analysis and Manipulation," *ACM Trans Graphics,* vol. 23, pp. 368-376, 2004.
[30] C. Ma, L.-Y. Wei, and X. Tong, "Discrete Element Textures," *ACM Trans Graphics,* vol. 30, pp. 62:1-62:10, Aug. 2011.
[31] W. Matusik, M. Zwicker, and F. Durand, "Texture Design Using a Simplicial Complex of Morphable Textures," *ACM Trans Graphics,* vol. 24, no. 3, pp. 787-794, 2005.
[32] P. Pérez, M. Gangnet, and A. Blake, "Poisson Image Editing," *ACM Trans Graphics,* vol. 22, pp. 313-318, 2003.
[33] T. Porter and T. Duff, "Compositing Digital Images," *ACM SIGGRAPH,* vol. 18, no. 3, pp. 253-259, 1984.
[34] G. Martens, C. Poppe, P. Lambert, and R.V.d. Walle, "Noise- and Compression-Robust Biological Features for Texture Classification," *The Visual Computer,* vol. 26, no. 6, pp. 915-922, 2010.
[35] E. Risser, C. Han, R. Dahyot, and E. Grinspun, "Synthesizing Structured Image Hybrids," *ACM Trans Graphics,* vol. 29, pp. 85:1-6, 2010.
[36] L. Ritter, W. Li, M. Agrawala, B. Curless, and d. Salesin, "Painting with Texture," *Proc. 17th Eurographics Symp. Rendering (EGSR),* pp. 371-377, 2006.
[37] A. Rosenberger, D. Cohen-Or, and D. Lischinski, "Layered Shape Synthesis: Automatic Generation of Control Maps for Non-Stationary Textures," *ACM Trans Graphics,* vol. 28, no. 5, p. 107, 2009.
[38] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts," vol. 23, no. 3, pp. 309-314, 2004.
[39] C. Steger, "An Unbiased Detector of Curvilinear Structures," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 20, no. 2, pp. 113-125, Feb. 1998.
[40] S. Todorovic and N. Ahuja, "Texel-Based Texture Segmentation," *Proc. IEEE 12th Int'l Conf. Computer Vision (ICCV),* pp. 841-848, 2009.
[41] M. Ding and R.-F. Tong, "Content-Aware Copying and Pasting in Images," *The Visual Computer,* vol. 26, no. 6, pp. 721-729, 2010.
[42] J. Wang and M.F. Cohen, "Image and Video Matting: A Survey," *Foundation and Trends in Computer Graphics Vision,* vol. 3, pp. 97-175, 2007.
[43] L.Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the Art in Example-Based Texture Synthesis," *Eurographics, State of the Art Report,* 2009.
[44] L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. SIGGRAPH,* pp. 479-488, 2000.
[45] L. Wei, "Texture Synthesis from Multiple Sources," *Proc. ACM SIGGRAPH Sketches and Applications,* p. 1, 2003.
[46] Q. Wu and Y. Yu, "Feature Matching and Deformation for Texture Synthesis," *ACM Trans Graphics,* vol. 23, no. 3, pp. 364-367, 2004.
[47] J. Zhang, K. Zhou, L. Velho, B. Guo, and H. Shum, "Synthesis of Progressively-Variant Textures on Arbitrary Surfaces," *ACM Trans Graphics,* vol. 22, no. 3, pp. 295-302, 2003.
[48] D. Aiger, D. Cohen-Or, and N.J. Mitra, "Repetition Maximization Based Texture Rectification," *Computer Graphics Forum,* vol. 31, no. 2, 2012.

Fang-Lue Zhang received the BS degree from the Zhejiang University in 2009. Currently, he is working toward the PhD degree in Tsinghua University. His research interests include computer graphics, image processing and enhancement, image and video analysis and computer vision.

Ming-Ming Cheng received the BS degree from the Xidian University in 2007. Currently, he is working toward the PhD degree in Tsinghua University. His research interests include computer graphics, computer vision, image processing, and sketch-based image retrieval. He has received a Google PhD fellowship award, an IBM PhD fellowship award, and a "New PHD Researcher Award" from the Chinese Ministry of Education. He is a member of the IEEE.

Jiaya Jia received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2004 and is currently working as an associate professor in the Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK). He was a visiting scholar at Microsoft Research Asia from March 2004 to August 2005 and conducted collaborative research at Adobe Systems in 2007. He leads the research group in CUHK, focusing specifically on computational photography, 3D reconstruction, practical optimization, and motion estimation. He serves as an associate editor for TPAMI and as an area chair for ICCV 2011. He was on the program committees of several major conferences, including ICCV, ECCV, and CVPR, and cochaired the Workshop on Interactive Computer Vision, in conjunction with ICCV 2007. He received the Young Researcher Award 2008 and Research Excellence Award 2009 from CUHK. He is a senior member of the IEEE.

Shi-Min Hu received the PhD degree from Zhejiang University in 1996. Currently, he is working as a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing. He is an associate editor-in-chief of The Visual Computer, associate editor of Computer and Graphics and on the editorial board of Computer Aided Design. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is a member of the IEEE.