Nested Images

Qiang Tong[#1], Song-Hai Zhang[#2], Ralph R. Martin[*3], Paul L. Rosin[*4]

[#]*Tsinghua National Laboratory for Information Science and Technology,*
Department of Computer Science and Technology, Tsinghua University
Beijing, P. R. China
[1]tong-q04@mails.tsinghua.edu.cn
[2]shz@tsinghua.edu.cn

[*]*School of Computer Science & Informatics, Cardiff University*
Cardiff, United Kingdom
[3]ralph@cs.cf.ac.uk
[4]Paul.Rosin@cs.cardiff.ac.uk

Abstract—**A *nested image* is a form of artistic expression in which one or more secondary figures are embedded within a primary figure one by one. Contours of the primary figure, especially the contours of its inner holes, are used to portray a secondary figure. The secondary figure is totally inside the primary figure, and it would use some inner holes of primary figure as part of itself, which would produce an artistic effect. Here, we present a system for creating such images. It relies on the experience of human perception to minimize the difference between embedded inner figure and primary outer figure. Our system detects the enclosed outer contour of the figure to be nested, and then finds a place in the outer figure to embed it, together with a suitable transformation for doing so, by optimizing an energy based on the distance of contours. Morphing is done by an iterative approach, which warps the corresponding contours of the inner figure and the holes of the outer figure to an appropriate position. We show various nested images generated by our system.**

***Keywords*—Nested Image, Shape Matching, Contour Distance, Image Morphing**

I. INTRODUCTION

Nested image is a special style for figures (see Fig. 1), especially for clip art, in which several figures are nested one by one, and the holes of the outer figures are used as parts of the inner nested figures. To build a nice nesting artwork by artists, the outer and inner figures are designed carefully, so that they can match perfectly. From observation of the nested image style of artwork, it was found that the holes of the outer figures are overlapped on the end of the inner figures, so that they become meaningful parts of the nested image, usually as its end-parts, i.e., in Fig. 1 the hole of the outer figure performs the role of a handkerchief of the inner figure.

Even for artists it is difficult to nest two arbitrary given figures, due to mismatches between the shapes. So we give an algorithm for automatically generating *nested images* based on a ClipArt library. While the user gives an outer figure, the algorithm can automatically search for the best matched nested image and its nesting posture (including the scale, rotation, and position). It starts by extracting the contour (both outer and inner) of the outer figure and the enclosed outer contour of the inner figure to be nested, and then searches for an appropriate position in the outer figure at which to embed a transformed version of the inner figure, by following several restrictions we present.

Fig. 1 Some nested images. Left: using clip art images, while the hole of most outer figure is used as a handkerchief of inner figure. Right: the tree is used for other image

When candidate figures are selected they are still not perfectly matched. We morph the outer figure and the nested one by minimizing the distortion in order to match the nested figure and the holes of the outer figure. Morphing is done using an iterative approach, by moving the contour of the inner holes of the outer figure and the outer contour of the inner figure, completing the nested image. We demonstrate our system with several examples, showing that it is potentially a useful creative tool for artists.

II. RELATED WORKS

Several image processing methods have been proposed to offer fancy tools and generate artistic effects based existing images from image library, such as image zoomquilts [4] and image based painterly rendering [8]. Our method also find good nested images by searching good candidates from an image library.

Some previous works have considered how to compose a background image by several small figures [17]. The closely related works are the various mosaicing algorithms that place tiles into a background mask under different restrictions.

Photomosaics construct a big image from a collection of small figures arranged in a rectangular grid [6], [14]. For each rectangular block of pixels in the input background image, the photomosaic algorithm searches a database of tiles to find the most similar one of the original block. Simulated Decorative Mosaic [7] align square tiles with varying orientations to preserve the edges of input background image while covering most area by the colored tiles. Jigsaw Image Mosaic [10] composes images with tiles of arbitrary shape by minimizing a mosaicing energy function. These approaches focus on filling the region of a background image with several small images or tiles.

Fig. 2 Some related works. Left: Photomosaic. Middle: Simulated Decorative Mosaic. Right: Jigsaw Image Mosaic

Some other works also focus on how to hide objects or figures. Yoon et al. [18] apply a stylized line drawing method to both background image and object images, to find suitable places to hide small and simple objects, generating hidden-picture puzzles. Mitra et al. [13] generate emergent images of 3D objects using a synthesis method which enables objects to be easily detected by humans, but which are much more difficult for an automatic algorithm to recognize. Chu et al. [5] present a texture synthesis technique for generating camouflaged images that use the object segmentation and their topological relations as clues for humans to detect. These approaches tend to hide objects in the background image to prevent people from detect these objects easily. However, we prefer to reveal the embedded object by inverting its color.

Shape matching in images broadly uses either brightness-based methods or feature-based methods [15]; several variants exist of each approach. Brightness-based methods treat the intensity of each pixel in the image as the feature descriptor, although these values may be affected by factors such as the poses of objects and illumination changes [3], [16]. On the other hand, feature-based methods describe the shapes of objects in an image in the spatial domain, using a much smaller number of features such as the well-known SIFT descriptors [12]. Other papers use the relations between contours in an image as shape descriptors [9], [11]. The shape context method [1] is translation and scale invariant, and has been widely used in shape matching, especially for character recognition. [2] introduces geometric blurring for robust template matching under affine distortion. Most of these methods consider the similarity between two full shapes, however, in our case, we have to find the most similar matching on part of the shape.

III. Approach

We present an automatic system to generate a *nested image*. We define a nested image as a new image which combines an outer clip art figure with an inner clip art figure; the latter are embedded in such a way that its color is opposite from the former. Thus people can recognize the outer object when they pay attention to one color, and recognize the inner object when paying attention to another color. Here, for simplicity, we only consider the two-level case, that is, one figure is nested into the other. The multiple levels case as Fig. 1 shows can be done by repeated application of our method. The algorithm is composed of two steps. The first is a search for candidate figures from a ClipArt library, which finds the best matched inner figure for the outer object, measured in terms of differences in similarity between the outer contour of the inner object and contours of the outer object. The second is image morphing to fit the parts of the inner object and the shape of the holes of the outer object together.

The input to our system is just a clip art A as the outer figure, in which there should have at least one hole. The inner object B which is to be nested into it is chosen from a clip art library automatically by our algorithm. The construction of this clip art library will be discussed later. We start by extracting all contours c(A) of A, and extracting the contours $C(H_i)$ $(i = 1, \cdots, m)$ of inner holes H_i if A contains m holes. Denote the enclosed outer contour of A as C(A), we have

$$c(A) = C(A) + \sum_{i=1,\cdots,m} C(H_i).$$

The first step when generating a nested image is to find the best fitting clip art object B for embedding in A. We use a candidate figure searching approach to solving this problem,

Fig. 3 Clip art figures library

based on measuring the similarity between every H_i and subpart of B from a library. It finds a candidate figure as well as the location for it in which H_i would be similar as its outer contour. Then we apply an iterative morphing approach to perform figure embedding. In every iteration of approach, the corresponding parts of B and H_i would be approaching to each other, until the similarity between their shapes becomes acceptable.

We construct a database of 400 Clip Art figures, as well as their mirror figures. All figures in the library are unified in the same height 1024. Fig. 3 shows some example clip art figures from our library.

IV. Candidate Figure Search

We use an image based method for candidate figure search. Suppose figure A is the given outer figure, and figure B_j is a figure in the library which we have described in the last section.

As we mentioned previously, we have $c(A) = C(A) + \sum_{i=1,\cdots,m} C(H_i)$. The candidate figure B_j would be under a certain similarity transformation to achieve the transformed figure B_j^T for similarity measurement, and we use $C(B_j^T)$ to illustrate the enclosed outer contour of B_j^T. The similarity transform is defined as below,

$$B_j^T(x,y) = \begin{pmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{pmatrix} B_j(x,y) + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

where s is a scale factor, θ is a rotation angle, while Δx and Δy represent a translation.

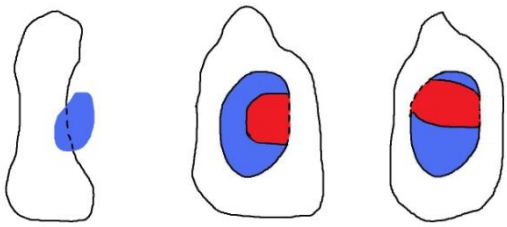
We define $\mathbb{E}$ to measure the similarity between A and B_j^T. In our implementation, we match the figure B_j^T and A to find the minimal $\mathbb{E}$ for θ in every 15°and s from 0.2 to 0.5 with an increment of 0.1 and iterating Δx and Δy pixel by pixel so that C(B) is totally inside of C(A). Iterating Δx and Δy pixel by across the outer figure while keeping the transformed object within the host figure would be very costly. One directly idea is applying some method based on fluency, however, we don't need the whole figure of C(B) be similar to C(A) but just subpart of C(B) to fit some of $C(H_i)$. Instead, we apply the candidate figure searching at a coarser scale. Since every figure in our clip art figures library are 1024 height, we construct the image pyramid for every figure, use their 256 height version for the searching. This would reduce computation time markedly. After the searching have been done at coarser scale, it would propagate down to the finer scale which is the upsampled version with a factor of 2 of coarser scale.

Similarity $\mathbb{E}$ is defined as below,

$$E = \arg\max_{B_j^T} h(C(B_j^T), C(A)) \prod_{i=1}^{m} h(C(H_i), C(B_j^T))$$

h(M, N) is defined as the one-way Hausdorff distance from a point set M to the other point set N, which means we just find the nearest point $q \in N$ for every $p \in M$, but not the opposite. Moreover, we use a weight strategy for computing Hausdorff distance. We label the points on C(A) with a weight 1.0 while using 0.5 for those points on $C(H_i)$. By this, the computed Hausdorff distance would tend to close to $C(H_i)$, which is the inner contour of figure A.

The algorithm proposes finding a place that at least one hole fits the contour of B_j^T. However, Fig. 4 shows some failure cases that against these three restrictions.



(a) (b) (c)

Fig. 4 Some fail cases. Blue region indicates B, while red region indicates H_i. (a): B is outside of C(A). (b): The length of C(B) would be longer while subtract H_j from B. (c): The number of connected regions of B increases while subtract H_j from B.

With these failure cases, we restrict the holes for the better result for the following morphing process. By subtracting hole region H_i from B_j^T, the length of outer contour of B_j^T should not become longer, and the number of connected regions should not increase. Thus we summarize three restrictions for candidate figure searching:

- The nested figure B_j^T are totally in the C(A)
- The length of $C(B_j^T)$ should be smaller than $C(B_j^T - H_i)$, where $B_j^T - H_i$ means the subtraction H_i from B_j^T
- The number of connected regions of B_j^T should be no smaller than $B_j^T - H_i$

Fig. 5 Some matching example cases. Left: Input outer figures and candidate inner figures. Middle: Simply copy candidate figures into outer figures. Right: Bigger version of matching results.

Fig. 5 shows some candidate figures for several outer figures, which are chosen automatically from our library under the restrictions. We use B to denote the best candidate inner figure for an outer figure A.

V. MORPHING

Usually, the best matched candidate figures from the figure library still do not perfectly match with the outer figure. We need to apply a morphing process to deform the contour of holes and the candidate figure to fit together.

Since A may contains several holes, firstly we determine which holes would be deformed as part of B. By simply pasting B onto A, we compute the overlapping area rate of every $H_i \subset A$. If H_i is overlapped by more than 90%, we record it as one of H_k. We use blue lines to indicate C(B), and use red lines to indicate $C(H_k)$, as Fig. 6(a) shows.

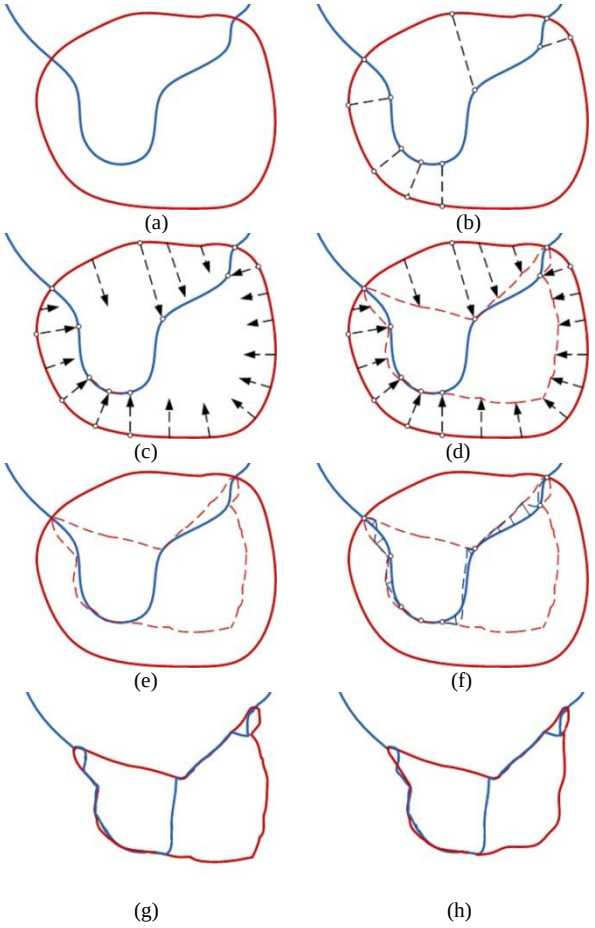


Fig. 6 Iterative morphing process.

For each pixel $p \in C(H_k)$, we find its nearest point $q \in C(B)$, and then we find the nearest point $p' \in C(A)$ of q. If p and p' are the same point, thus we record $p \in C(H_k)$ and $q \in C(B)$ are one pair. The circles on each lines connected by dashed lines as Fig. 6(b) shows indicates these one-to-one nearest point pairs.

As we mentioned in the previous section, H_k should appear at end point of B. Thus we also need to figure out which part of B would be deformed to H_k. Then we apply an iterative method as Algorithm I indicated to achieve the morphing effect.

ALGORITHM I
CONTOUR MORPHING

while $S(H_k \cap B)/\, S(H_k) < thres$, **do**
  $H'_k \leftarrow \mathrm{Morph}(H_k,\ B)$
  $B \leftarrow \mathrm{Morph}(B \cap H'_k,\ H'_k)$
  $H_k \leftarrow H'_k$, $B \leftarrow B'$
end while
return B

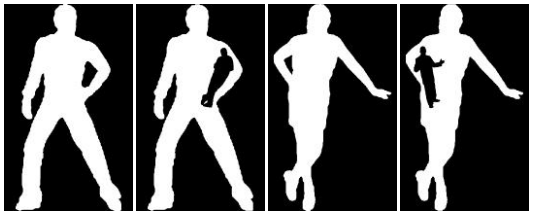
ALGORITHM III
FUNCTION MORPH (M, N)

For all $p \in C(M)$, **do**
  $q \leftarrow \mathrm{FindNearestPoint}(p, C(N))$
  $p' \leftarrow \mathrm{FindNearestPoint}(q, C(M))$
  if $p = p'$ **then**
    $v(M(p) \leftarrow q - p$
  end if
end for
$v(M) \leftarrow \mathrm{VectorField}(M, v(M))$
$M' \leftarrow M + v(M)$
return M'

For the point pair composed by p and q, we compute the vector of p by $v(p) = q - p$. Then we spread the vector field on the whole region of H_k by interpolation method as Fig. 6(c) shows, which would morph H_k as B. For every $p \in H_k$, we find its destination pixel $p' = p + v(p)$, this would produce a new region H'_k, see Fig. 6(d). While we produce the morphed hole region H'_k as Fig. 6(e) shows, we repeat the morphing process, however, this time it is applied to B as blue dashed lines in Fig. 6(f). Note that the vector fields are interpolated in region $B \cap H'_k$. This process would cause the figure region morphs as the hole, see Fig. 6(g). Repeat above steps iteratively, Fig. 6(h) shows one more iteration, we could morph B and H_k as similar as possible. When H'_k is covered by B by a predefined amount the morphing process terminates.

VI. RESULTS

In Fig. 7 and Fig. 8 we show several examples created by our system, which are produced using an Intel Core2 2.10GHz PC with 2GB memory. It takes 1.5 seconds to match and morph a figure pair with 1024 pixels height and about 512 pixels width. With a given outer figure, it takes about 5 minutes to 30 minutes, which depends on the area of the given outer figure, to search the candidate inner figure from the whole library we constructed and generate the morphing result.



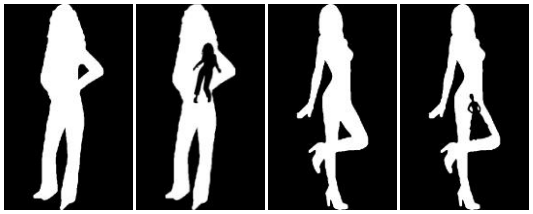


Fig. 7 Some results. Left: Input primary outer figure with holes. Right: Nested images created by our system.

The results shown in Fig. 7 focus on primary outer figures with holes, moreover, our algorithm also apply on those primary outer figures without inner holes. Fig. 8 shows some examples. In that case, it would degenerate to measuring the one-way Hausdorff distance between the inner figure and outer figure, with those restrictions still. Our algorithm can be applied several times to produce a nested image with multiple levels of inner figures as Fig. 9 shows. Moreover, the user could apply this approach several times to produce the one primary outer figure with multiple inner figures in one level, just as Fig. 10 shows, however, our approach is not designed for filling a primary figure with multiple figures, thus it would produce a figure embedded with several non-overlapped figures but not be fully covered as Fig. 2.

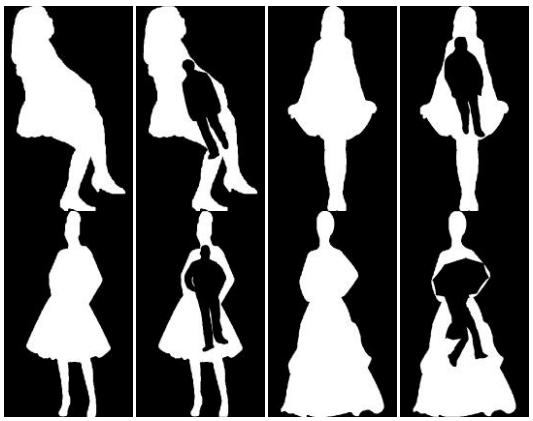


Fig. 8 Some results. Left: Input primary outer figure without holes. Right: Nested images created by our system.

Fig. 9 Some results with multiple levels of inner figures. Left: Input primary outer figures. Right: Nested image created by our system.

Fig. 10 Some results with multiple inner figures in one level. Left: Input primary outer figures. Right: Nested images created by our system.

In our experiments, we notice that the results are always aesthetically unacceptable if the inner figure is positioned at the head of the outer figures. This phenomenon implies that we could produce results by adding some semantic restrictions, for example, to restrict the nested figure to be totally in the body but not to cover the whole of the outer figure.

VII. Conclusion and Future Work

This paper has given an automatic system for generating *nested images*. By applying techniques for contour matching and shape transformation, our system can produce pleasing results which embed partly transformed an inner object figure into an outer figure by considering its holes. Viewers can see the embedded object easily because it has the opposite color from outer figure, while some holes of the outer figure would be part of the inner figure. Although our method cannot substitute for the creative work of artists, it can produce good results by matching the existing figures in the ClipArt library. We believe our approach can provide an effective tool for amateurs to create nested images, or for skillful artists to plan their designs before execution.

Some further works could improve our system. Our result is relying on the matching degree of inner and outer figures, so it can be improved by increasing the library size or using internet image search. Our approach applies the Hausdorff distance to measure the difference between contours of outer figure and inner figure, especially the difference between inner figures and holes of outer figure. [18] presents a simple rotation-invariant method for defining shape contexts, which may be considered to improve the effect of matching between inner figures and holes of outer figure. As it compares the distributions of sampled points, its effectiveness depends on how to choose sampled points in a complex and large figure. Since we just need to find a part of contour of figure be matched well but not the whole figure, we plan to investigate variants of the shape context method for measuring similarity between sub-parts of two figures but not whole figures. We also hope to further improve our system by considering other recent progress in human perception.

Acknowledgment

This work was supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Number 60970100, 61033012).

REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[2] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proc. Computer Vision and Pattern Recognition*, pp. 26–33, 2005.

[3] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models: their training and application," *Computer Vision and Image Understand*

[4] -F. Tong, and J.-X. Dong, "Making Slide Shows with Zoomquilts, *ing*, vol. 61, no. 1, pp. 38–59, 1995.

[5] L. Cong, R." *Journal on Computer Science and Technology*, vol. 25, no. 3, pp. 572–582, 2010.

[6] H.-K. Chu, W.-H. Hsu, N. J. Mitra, D. Cohen-Or, T.-T. Wong, and T.-Y. Lee, "Camouflage Images," *ACM Transactions on Graphics*, vol. 29, no. 3, p. 51, 2010.

[7] A. Finkelstein, M. Range, "Image Mosaics," *Artistic Imaging and Digital Typography*, LNCS, vol. 1375, 1998.

[8] A. Hausner, "Simulating Decorative Mosaics," in *Proc. of SIGGRAPH*, pp. 573–580, 2001.

[9] H. Huang, Y. Zang, and C.-F. Li, "Example-based painting guided by color features," *Visual Computer*, vol. 26, no. 6-8, pp. 933–942, 2010.

[10] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[11] J. Kim, and F. Pellacini, "Jigsaw image mosaics," in *Proc. of SIGGRAPH*, pp. 657–664, 2002.

[12] L. J. Latecki, R. Lakamper, and U. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Proc. Computer Vision and Pattern Recognition*, pp. 424–429, 2000.

[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 20, pp. 91–110, 2003.

[14] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or, "Emerging images," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1–8, 2009.

[15] R. Silvers, M. Hawley, *Photomosaics*, Henry Holt, New York, 2000.

[16] R. C. Veltkamp, and M. Hagedoorn, "State of the art in shape matching," *Principles of visual information retrieval*, Springer, pp. 87–119, 2001.

[17] T. Vetter, M. J. Jones, and T. Poggio, "A Bootstrapping Algorithm for Learning Linear Models of Object Classes," in *Computer Vision and Pattern Recognition*, pp. 40–46, 1997.

[18] J. Xu, C. S. Kaplan, "Calligraphic Packing," in *Proc. Graphics Interface*, pp. 43–50, 2007.

[19] J. Yoon, I. Lee, and H. Kang, "A Hidden-picture Puzzles Generator," *Computer Graphics Forum*, vol. 27, pp. 1869–1877, 2008.