

Nested Images

Qiang Tong *, Song-Hai Zhang

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Ralph R. Martin, Paul L. Rosin

School of Computer Science & Informatics, Cardiff University, Cardiff, United Kingdom

Abstract

A *nested image* is a form of artistic expression in which one or more secondary figures are embedded within a primary figure, perhaps recursively. Contours of the primary figure are used to contain a secondary figure; the effect has a particularly interesting artistic effect if parts of the secondary figure have a corresponding shape to inner holes of the primary figure. Here, we present a system for creating such images. 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 between the contours. We also allow small changes of shape, which can help to match contours. Morphing is done iteratively, warping the corresponding contours of the secondary figure and the holes of the primary figure to appropriate positions. We show various nested images generated by our system.

Key words: Nested Images, Shape Matching, Contour Distance, Image Morphing

1. Introduction

A *nested image* is a particular style of artwork, often but not exclusively based on silhouettes, in which an inner figure is nested in an outer figure, perhaps recursively. The effect has a particularly interesting artistic effect if certain holes within an outer figure correspond to certain extremities of an inner figure, in a meaningful way—for example, in Fig.1, the inner orange woman is holding a handkerchief, forming the gap between the outer woman’s arm and body. To produce an aesthetic result, the outer and inner figures must be designed carefully to ensure a good match.

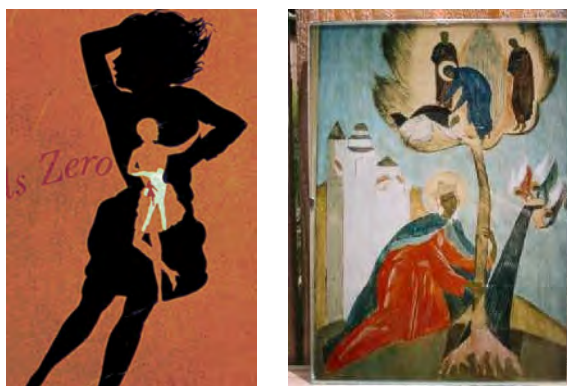


Figure 1. Examples of nested images. Left: recursively nested silhouettes; the inner orange woman holds a handkerchief, forming the gap between the outer woman’s arm and body. Right: the crown of the tree contains an embedded image.

Even artists find it difficult to choose images for nesting, due to the difficulty in finding or planning matches between the shapes. Thus, here we give an algorithm for automatically generating *nested images*

*Corresponding author email: tong-q04@mails.tsinghua.edu.cn

using a clip-art library of silhouettes as source material. The user selects an outer figure, after which the algorithm automatically searches for and determines the one or more best matching nested images and their nesting transform (including scaling, rotation, and translation). This is done by extracting the contours of the outer figure (both outer and those of any holes), and the outer contour of each candidate inner figure, and then searching for an appropriate pose within the outer figure at which to embed a transformed version of the inner figure; we discuss restrictions placed on this pose later.

Even after candidate figures have been selected, the result is nearly always an imperfect match. We thus morph both the outer and inner figures to match the part of the inner figure with the hole in the outer figure, while minimizing the distortion. This is done using an iterative approach.

We demonstrate our system with several examples, showing that it is potentially a useful creative tool for artists.

2. Related Work

Several computer graphics methods have been proposed which offer tools to help generate artistic effects using existing images from an image library, two particular examples of which are image zoomquilts [4] and image based painterly rendering [8]. Our method is also based on use of suitable images from an image library.

Various previous works have considered how to compose an overall image from several smaller figures or tiles [17]. For example, several mosaicing algorithms exist which place tiles within a shape mask under various constraints. The *photomosaics* technique constructs a large image from a collection of small images 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 best matching the original block. The *simulated decorative mosaic* approach [7] aligns square tiles with varying orientations to preserve edges in an input image while covering its area (apart from small gaps) by appropriately colored tiles. The *jigsaw image mosaic* approach [10] composes images from tiles of arbitrary shape by minimizing a mosaicing energy function. All of these methods typically work with many, smaller, subimages than our inner images.

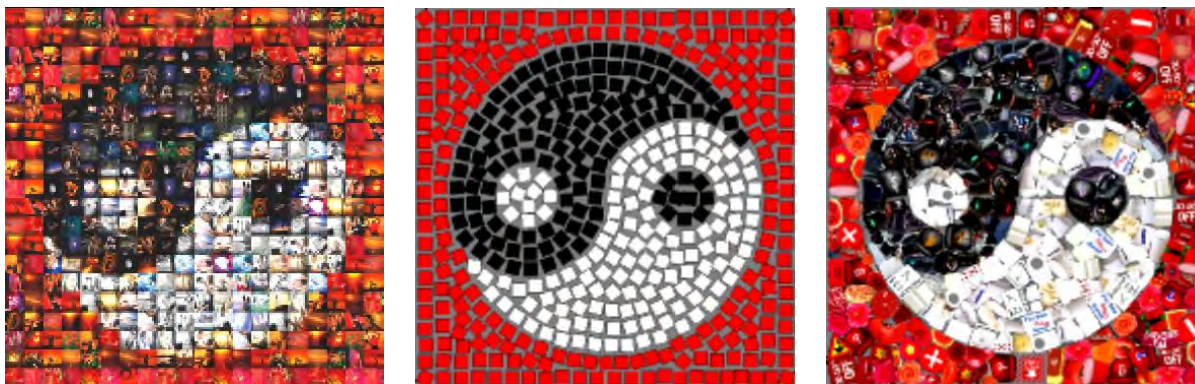


Figure 2. Related works. Left: Photomosaic. Middle: Simulated Decorative Mosaic. Right: Jigsaw Image Mosaic

Other work has investigated how to hide objects or figures within background images. Yoon et al. [18] generate *hidden-picture puzzles* by applying a stylized line drawing method to both the background image and object images, which allows them to find suitable places at which to hide small and simple objects. 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 object segmentations and their topological relations to hide an image within another. While these approaches concentrate on hiding objects, our goal has strongly contrasting nested objects for visibility.

Shape matching methods in images broadly use either a brightness-based or feature-based approach [15]; several variants exist of each. *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 rely on 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 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 for shape matching, especially for character recognition. *Geometric blurring* [2] has also been used for robust template matching under affine distortion. Most of these methods consider the similarity between entire shapes, whereas we need to match part of one shape to another.

3. Approach

3.1. Overview

We now present our automatic system for generating a *nested image*, which we define for this purpose to be a new image which combines an outer silhouette with an inner silhouette; the latter is embedded in such a way that its color is opposite that of the former (typically the opposing colours are black and white). Thus people can recognize the outer object when they concentrate on one color, and the inner object when concentrating on the opposite color. Here, for simplicity, we only consider the two-level case, where one figure is nested into the other. Multiple levels of nesting as in Figure 1(left) can be achieved by repeated application of our method.

Our algorithm comprises two steps, given an input outer image. The first is a search for candidate figures from an image library, to find the inner figure which best matches the outer object, measured in terms of differences between the outer contour of the inner object and contours of the outer object. The second step performs image morphing, adjusting matching parts of the inner object and corresponding holes of the outer object to give a best fit.

The input to our system is just a single silhouette A as the outer figure, which should have at least one hole. The inner object B which is to be nested into it is chosen from a library of silhouettes automatically by our algorithm; the information stored with this library is discussed later. We start by extracting the set $c(A)$ of all contours of A , including the outer contour $C(A)$ and, assuming A contains m holes, the contour $C(H_i)$, $i = 1, \dots, m$ of each inner hole H_i . Thus

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

The first step when generating a nested image is to find the best fitting library object B for embedding in A . We compare candidates based on the similarity between every H_i and subparts of B . This both finds a matching figure, as well as the location at which H_i is similar to part of its outer contour. In the second step we apply an iterative morphing approach to perform figure embedding. In every iteration, the corresponding part of B is made more similar to H_i and vice versa.

3.2. Silhouette Library

We have constructed a database of 400 silhouette figures, and have also included their reflections about a vertical line. All figures in the library are normalised to a height of 1024 pixels. Figure 3 shows some examples from our library.



Figure 3. Silhouette library—examples.

4. Matching

We use an image based method for selection of an appropriate library figure for nesting. Suppose A is the given outer figure, and B_j is a library figure. We allow the candidate figure B_j to undergo a similarity transformation before matching; let B_j^T denote this transformed figure, and $C(B_j^T)$ its outer contour. We may determine B_j^T from B_j using

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

where s is a scale factor, θ is a rotation angle, and Δx and Δy represent a translation. We define an energy E to measure the similarity between A and B_j^T , as explained below; the goal is to minimise E . In our implementation, we measure similarity between A and B_j^T for $\theta = 0$ and at 15° intervals, s from 0.2 to 0.5 with an increment of 0.1, and with Δx and Δy stepping pixel by pixel so that $C(B)$ remains totally inside $C(A)$. Note, however, that we do not wish the whole of $C(B)$ to match $C(A)$ but just some subpart of $C(B)$ to one particular $C(H_i)$.

Iterating Δx and Δy pixelwise across the outer figure while keeping the transformed object within the host figure would be very costly. For efficiency, we apply candidate object search at a coarser scale. Every figure in our library has uniform height 1024. We construct an image pyramid for every figure, and use a subsampled figure of height 256 to perform an initial search, reducing the computation time markedly. Promising results at the coarser scale are reconsidered at a finer scale with height 512 before final matching.

The energy E used to measure similarity is defined as

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

where $h(M, N)$ is the one-way Hausdorff distance from a point set M to another 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 while computing Hausdorff distance, giving points on $C(A)$ a weight 1.0 while using 0.5 for points on $C(H_i)$. This causes the computed Hausdorff distance to tend to close to $C(H_i)$, which is the inner contour of figure A .

Our algorithm is intended to find a place where at least one hole fits the contour of B_j^T . However, if we were to use the algorithm directly, at times it would give results which differ from those intuitively wanted or expected, as shown in Figure 4. To avoid such problems, we place restrictions on holes when considering

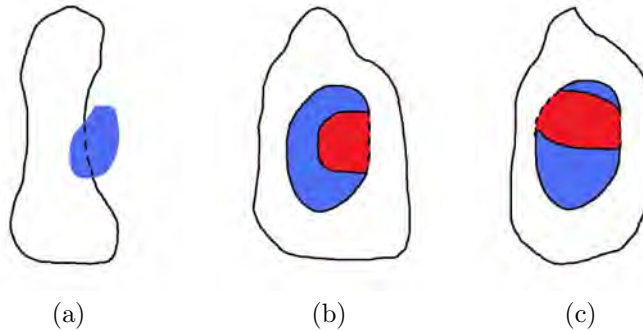


Figure 4. Unwanted results when using the unrestricted algorithm. Blue: B_j^T , red: H_i , black curve: $C(A)$. (a): B lies partly outside $C(A)$. (b): The length of $C(B)$ is made longer by subtracting H_j from B . (c): The number of connected regions of B is increased by subtracting H_j from B .

candidate solutions, as summarized below:

- The nested figure B_j^T must lie entirely within $C(A)$.
- The length of $C(B_j^T)$ should be smaller than $C(B_j^T - H_i)$, the contour left after subtracting H_i from B_j^T .
- The number of connected regions in $B_j^T - H_i$ should be no greater than in B_j^T

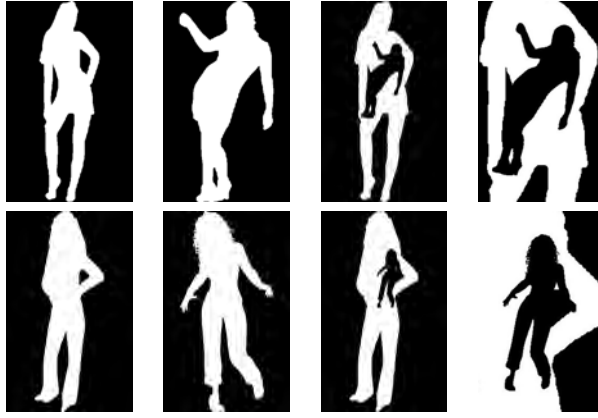


Figure 5. Some matching example cases. Left two: Input outer figures and candidate inner figures. Middle: Simply copy candidate figures into outer figures. Right: Bigger version of matching results.

Solutions not meeting these restrictions are rejected.

Figure 5 shows some candidate inner figures for several outer figures, the former were chosen automatically from our library given these restrictions. From now on, let use B denote the best candidate inner figure for an outer figure A .

5. Morphing

Usually, the best matching inner figure still does not perfectly match the outer figure. We thus apply a morphing process to deform hole and inner figure to make them fit together better.

Since A may contains several holes, firstly we determine which holes must be deformed to match 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 the H_k where k indexes the overlapped holes.

As noted earlier, each H_k should appear at some extremity of B . Thus we also need to determine which part of B must be deformed to agree with H_k . We then apply an iterative method given in Algorithm 1 indicated to achieve the morphing effect.

Algorithm 1 Contour Morphing

```

while  $S(H_k)/S(H_k \cap B) < t$  do
   $H'_k \leftarrow \text{Morph}(H_k, B)$ 
   $B' \leftarrow \text{Morph}(B \cap H'_k, H'_k)$ 
   $H_k \leftarrow H'_k, B \leftarrow B'$ 
end while
return  $B$ 

```

When morphing, for each pixel $p \in C(H_k)$, we find its nearest point $q \in C(B)$, and then we find q 's nearest point $p' \in C(A)$. If $p = p'$, we record $p \in C(H_k)$ and $q \in C(B)$ as a pair. The ends of dashed lines in Figure 6(b) indicate these one-to-one nearest point pairs. For each point pair p and q , we compute the vector $v(p) = q - p$. We use this to define a vector field on the whole of H_k by interpolation as shown in Figure 6(c). Applying this field to H_k morphs it towards B : for every $p \in H_k$, we find its destination position $p' = p + v(p)$, giving a new region H'_k as shown in Figures 6(d), 6(e). After doing so, we repeat the morphing process, however, applying it this time to B : see Figure 6(f). In this case the vector fields are interpolated in region $B \cap H'_k$. This process moves B closer to the modified hole H'_k : see Figure 6(g). These steps are repeated iteratively—Figure 6(h) shows the result after a further iteration—until B and H_k are sufficiently similar.

Algorithm 2 Function Morph(M, N)

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

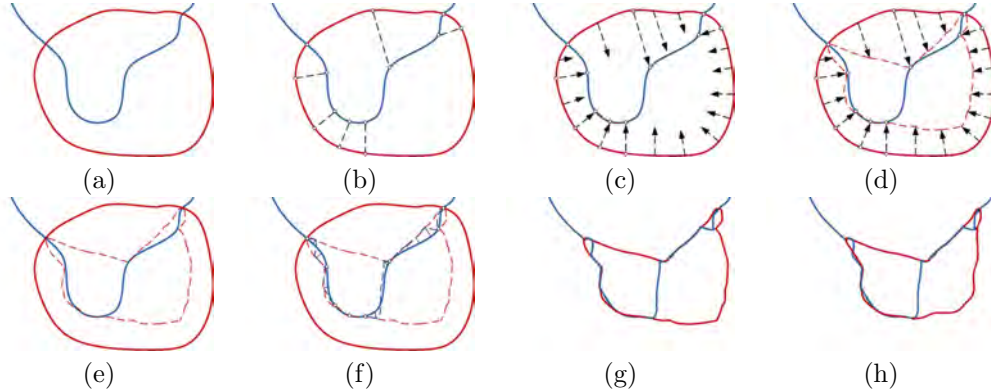


Figure 6. Iterative morphing process. Blue: $C(B)$, red: $C(H_k)$.

6. Results

Figures 7 and 8 show several examples created by our system, using an Intel Core2 2.10GHz PC with 2GB memory. It takes 1.5 seconds to match and morph a figure pair of size 1024×512 pixels. For a given outer figure, selecting a suitable inner figure from the whole library takes 5–30 minutes, depending on its area.

The results shown in Figure 7 consider outer figures with holes, the main focus of our algorithm. Figure 8 shows some examples without holes. In such cases, our approach degenerates to measuring the one-way Hausdorff distance between the inner figure and outer figure, under the given restrictions. Our algorithm can be applied several times to produce a nested image with multiple levels of inner figures as shown in Figure 9 shows. Moreover, the user could apply this approach several times to include multiple inner figures at the same level for a single outer figure, as shown in Figure 10. 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 Figure 2.

Our experiments, have lead us to conclude that the results are always aesthetically unacceptable if the inner figure is positioned within the head of the outer figures, and the inner figure should be approximately upright. The latter observation is why we restrict the roataion during search; further restrictions could be placed on matches to meet other aesthetic requirements.

7. Conclusions 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 a transformed and slightly deformed inner object figure into an outer figure by considering the latter’s holes. Viewers can see the embedded object easily because it has the opposite color from outer figure, while some holes of the outer figure become part of the inner figure. Although our method cannot substitute for the creative work of artists, it can produce good results by matching existing figures in a silhouette library. We believe our approach can provide an effective tool for amateurs to create nested images, or for skillful artists to plan potential designs

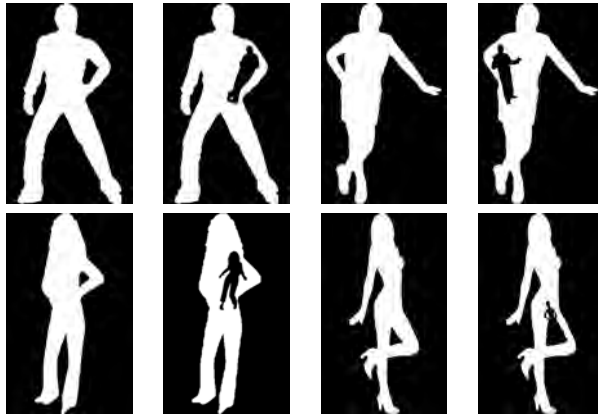


Figure 7. Some results. Left: Input primary outer figure with holes. Right: Nested images created by our system.

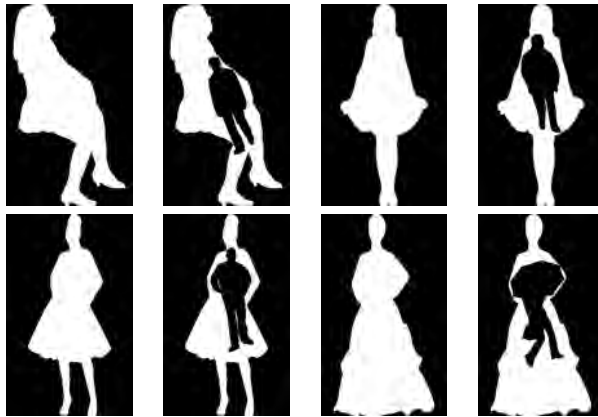


Figure 8. Some results. Left: Input primary outer figure without holes. Right: Nested images created by our system.

before execution.

Our system could be further improved. The quality of results could be improved by searching over a larger library, or even the Internet, at the cost of greater computation, of course. Our approach applies the Hausdorff distance to measure the difference between contours of the outer and inner figures, particularly holes of the outer figure. Yoon [18] presents a simple rotation-invariant method for defining shape contexts, which could be useful for such matching. As it compares the distributions of sampled points, its effectiveness depends on how to choose sampled points in a complex and large figure. We plan to investigate variants of this method applicable to partial figures.

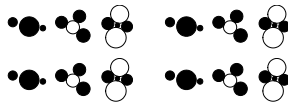


Figure 9. Some results with multiple levels of inner figures. Left: Input primary outer figures. Right: Nested image created by our system.

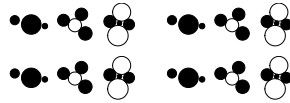


Figure 10. Some results with multiple inner figures in one level. Left: Input primary outer figures. Right: Nested images created by our system.

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* **24(4)** (2002), pp. 509–522.
- [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* (2005), pp. 26–33.
- [3] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, Active shape models: their training and application, *Computer Vision and Image Understanding* **61(1)** (1995), pp. 38–59.
- [4] L. Cong, R.-F. Tong, and J.-X. Dong, Making Slide Shows with Zoomquilts, *Journal on Computer Science and Technology* **25(3)** (2010), pp. 572–582.
- [5] 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* **29(3)** (2010), pp. 51.
- [6] A. Finkelstein, M. Range, Image Mosaics, *Artistic Imaging and Digital Typography, LNCS* **1375** (1998).
- [7] A. Hausner, Simulating Decorative Mosaics, in *Proc. of SIGGRAPH* (2001), pp. 573–580.
- [8] H. Huang, Y. Zang, and C.-F. Li, Example-based painting guided by color features, *Visual Computer* **26(6-8)** (2010), pp. 933–942.
- [9] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, Comparing images using the hausdorff distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15(9)** (1993), pp. 850–863.
- [10] J. Kim, and F. Pellacini, Jigsaw image mosaics, in *Proc. of SIGGRAPH* (2002), pp. 657–664.
- [11] 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* (2000), pp. 424–429.
- [12] D. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **20** (2003), pp. 91–110.
- [13] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or, Emerging images, *ACM Transactions on Graphics* **28(5)** (2009), pp. 1–8.
- [14] R. Silvers, M. Hawley, *Photomosaics*, Henry Holt, New York, 2000.
- [15] R. C. Veltkamp, and M. Hagedoorn, State of the art in shape matching, *Principles of visual information retrieval*, Springer, 2001, pp. 87–119.
- [16] T. Vetter, M. J. Jones, and T. Poggio, A Bootstrapping Algorithm for Learning Linear Models of Object Classes, in *Computer Vision and Pattern Recognition* (1997), pp. 40–46.
- [17] J. Xu, C. S. Kaplan, Calligraphic Packing, in *Proc. Graphics Interface* (2007), pp. 43–50.
- [18] J. Yoon, I. Lee, and H. Kang, A Hidden-picture Puzzles Generator, *Computer Graphics Forum* **27** (2008), pp. 1869–1877.