Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Guest Editors)

Volume 27 (2008), Number 7

A Hidden-picture Puzzles Generator

Jong-Chul Yoon,[1] In-Kwon Lee[1] and Henry Kang[2]

[1]Dept. of Computer Science, Yonsei University
[2]Dept. of Mathematics and Computer Science, University of Missouri, St. Louis

Abstract
A hidden-picture puzzle contains objects hidden in a background image, in such a way that each object fits closely into a local region of the background. Our system converts image of the background and objects into line drawing, and then finds places in which to hide transformed versions of the objects using rotation-invariant shape context matching. During the hiding process, each object is subjected to a slight deformation to enhance its similarity to the background. The results were assessed by a panel of puzzle-solvers.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computing Methodologies]: Computer Graphics-Applications J.5 [Computer Applications]: Arts and Humanities

1. Introduction

A hidden-picture puzzle consists of a background picture in which several extraneous graphical objects are hidden. To generate such a puzzle, an artist first sketches the background and the hidden objects, and then draw them as a coherent whole so that the hidden objects are not easily noticeable. Hidden-picture puzzles have been widely used for educational, commercial, and entertainment purposes, appearing regular items in newspapers and magazines, and on the web [Mal99]. Like many other types of puzzle, they not only provide amusement but also help improve the visual skills of the solvers.

According to Ball [Bal08], a well-known designer of hidden-picture puzzles, a range of skills are required to generate these puzzles, including shape recognition, spatial perception, eye coordination, memory, concentration, creativity and imagination. This motivates our development of an automatic puzzle generation system.

Figure 1 shows example of hidden-picture puzzles drawn by Ball which exhibit typical properties of hidden-picture puzzles:

- Image abstraction: Hidden-picture puzzles are typically line drawings (Figure 1(a)) or cartoons with simplified colors (Figure 1(b)).
- Shape similarity: The background must be complicated enough to hide the objects easily. Also, the shapes of the hidden object, after some geometric modification, must be similar to the adjacent region of the background.
- Shape modification: The transformations and deformations used to increase the shape similarity must not destroy the identity of the object.

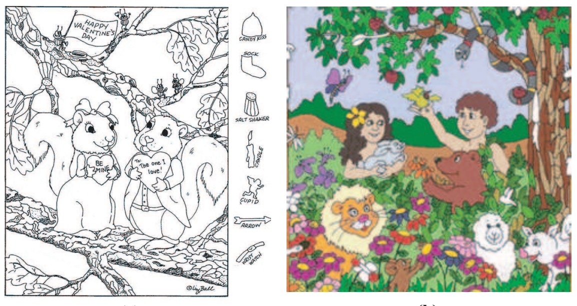


Figure 1: *Hidden-picture puzzles drawn by Ball (www.hiddenpicture.com): (a) hidden-picture puzzle in a line-drawing style; (b) colored hidden-picture puzzle with a limited number of colors.*

We have designed an automatic system to generate a hidden-picture puzzle (see Figure 2) that satisfies these properties. We start by converting photographs of potential hidden objects to line drawings and store them in an object database. The input image is also converted into a line drawing (or a colored abstraction). For each object in the database, the system searches an appropriate location in the background image at which to conceal it, using transformation-invariant shape matching. The system then selects the best matches between candidate objects and location. The objects with the best matches are then transformed, deformed and embedded in the background image to complete the puzzle.

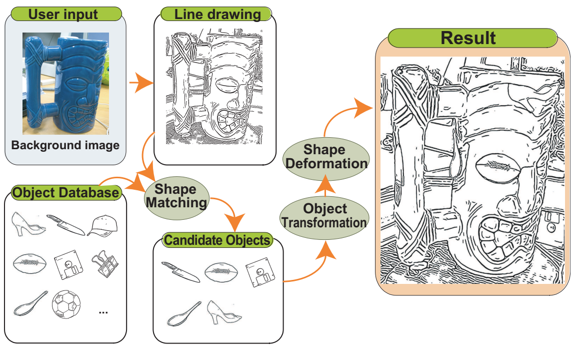


Figure 2: *System overview.*

2. Related Work

2.1. Image Stylization

Existing methods for image stylization mainly focus on aesthetic improvement of an image. For example, stroke-based rendering techniques [SWHS97, Her01, GCS02, DOM*01, HE04] re-renders image in a non-photorealistic style by filling its region with certain types of strokes. Image abstraction or tooning [DS02, WXSC04, CRH05, WOG06] provides stylization via image segmentation and line drawing. Our system adopts the cartoon-like abstraction style to imitate hand-drawn hidden picture puzzles. As an image puzzle generator, our work is also related to the image stylization algorithms that are driven by puzzle-like properties [KP02, XK07b, XKM07, XK07a].

2.2. Shape Matching

Veltkamp and Hagedoorn [VH01] suggest that the shape matching of images can be roughly classified into brightness and feature based methods. Brightness-based methods [CTCG95, VJP97] assume that the intensity of each pixel in the image is a feature descriptor, albeit modified by factors such as the poses of the objects depicted in the image and illumination changes. Feature-based methods recognize shapes in an image from the spatial configurations of a much small number of features. Lowe's SIFT descriptor [Low03] is well-known, and has been widely used in various applications. The relations among contours (or silhouettes) in an image can also be used as shape descriptors [HKR93, LLE00], and Berg et al. [BBM05] introduced geometric blur for robust template matching under affine distortion.

Because we need to determine the similarity between many possible regions in our background image and each hidden object in the object database, a fast matching method is required which is also transformation-invariant. We therefore selected the shape context method of matching [BMP02], which is simple but effective. As originally proposed, a shape context is translation- and scale-invariant; we have improved the method slightly by adding rotation-invariance (see Section 4.1).

3. Preprocessing Steps

3.1. Converting the Input Image to a Line drawing

We use the coherent line drawing (CLD) method by Kang et al. [KLC07] to generate coherent, smooth, and attractive line drawings of the background (Figure 3) and of the objects to be hidden.

The CLD method calculates the edge tangent vector flow, then uses the flow-based difference of Gaussians (FDoG) filter for edge detection, with iterative adjustments to enhance the continuity of edges. In our case, we apply CLD method on the input image to create the initial background image. Then after all the hidden objects are embedded into the background image, we apply CLD method again on the combined image to recalculate the edges. This re-processing of the edges makes the objects look more naturally blended in the background and less conspicuous. This is essential not only for the quality of final illustration but also for the difficulty of the puzzle.

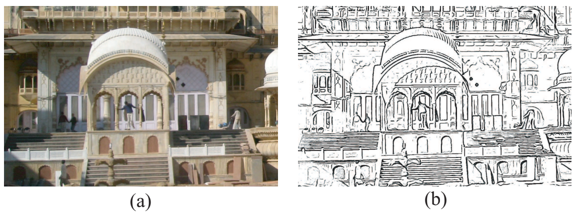


Figure 3: *(a) Input background image; (b) line drawing of the background.*

3.2. Object Database

We constructed a database of objects (to be hidden) from a set of arbitrary images. We convert these images into line-drawings using the CLD method. We then manually segment out the objects and put the line-drawing images of the segmented objects in the object database. In our experiments, we use the database of one hundred line-drawing images, each with a resolution of 300×300. To improve the likelihood of good matches, mirror images of the objects were added to the database.

4. Shape Matching

Consider a background image B in which one object O is to be hidden. We need to find a good match between a region in B and a copy of O that may be transformed by some combination of translation, rotation, and scaling. Exhaustive search of all possible configurations is infeasible. Instead, we compute transformation-invariant shape descriptors of O, and a sufficient numbers of candidate regions in B. The similarity between O and each candidate region in B can then be computed efficiently using the shape descriptors.

4.1. Rotation-Invariant Shape Context

The shape descriptors used in our system are based on the concept of shape context [BMP02], which abstracts the shape of a point cloud. The shape context consists of a small number of histograms which express the spatial relationships among the points representing the shape, and allow a possible match between two given shapes to be evaluated quickly. The use of directional relationships and relative distances between the points makes the shape context scale- and translation-invariant.

We first extract the feature points p_i, $(i = 1, \ldots, N)$ from the object image O using the well-known Harris corner detection method [HS98] (see Figure 4(b)). Then, N different shape contexts of O exist, each of which is computed with respect to each feature point. Considering a range of distances and directions from a specific center point p_i, the area covered by the bounding circle (with center p_i) of the point cloud can be subdivided into several bins (e.g, 32 bins in Figure 4(c)). The shape context of O with respect to the specific p_i is then determined by counting the number of points in each bin, which is effectively a two-dimensional histogram expressing the distance and direction of the other points from p_i.

One problem with the original version of shape context [BMP02] is that the histogram is not rotation-invariant. In other words, the shape contexts of two differently oriented version of an object are likely to be different, even though computed with respect to the same point. To obtain a rotationally invariant shape descriptor, we change the way in which we compute shape context by giving the directional bins a local basis. Figure 4(d) shows how we use principal components analysis (PCA) to extract the representative axis of the set of feature points, which is the best linear approximation of its shape. We recognize that the PCA-based representative axis may be undefined or poorly defined when an object is rotationally symmetrical, or nearly so; but most of the objects hidden in puzzles are not of this form, and in any case a nearly symmetric shape will not affect to the shape context because we use the same rule for the background as we do for the object. We use the representative axis to orientate eight directional regions. Combining these with four distance ranges, we obtain a rotation-invariant shape context histogram consisting of 32 bins (Figure 4(e)). We will use the notation $O_i(k)$, $(k = 1, \ldots, 32)$, to represent the number of points in the k^{th} bin of the histogram, where i is index of feature points in the object O.

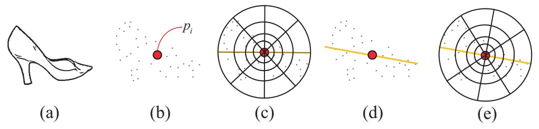


Figure 4: *A shape context of an object image for a specific feature point p_i: (a) object image in the database; (b) extracted feature points; (c) original (static) shape context of the object shape with 32 histogram bins; (d) representative axis computed by PCA; (e) rotation-invariant shape context.*

We can define rotation-invariant shape contexts of regions of the background image in a similar way. Feature points q_j, $(j = 1, \ldots, M)$ are again computed using Harris corner detection, but since the background image is usually much larger than the image of a hidden object, we can assume $M >> N$, where N is the number of feature points in the object image. We localize the shape context to each feature point q_j by considering only the N nearest neighbors of that feature point, as shown in Figure 5. We will use $B_j(k)$, $(k = 1, \ldots, 32)$ to denote the number of points in the k^{th} bin of the histogram, where j is index of feature points in the background image B.

4.2. Computing Similarity

We now compare all possible pairs of shape contexts B_j and O_i to find the best match, by calculating the similarity between each pair of shape contexts.

Since each shape context contains local shape information based on a central point, we can measure the local similarity of each pair of shape contexts (B_j, O_i) independently. A shape context is represented by an array of values, each of which corresponds to the number of points in a histogram bin. Therefore, we calculate the similarity between two shape contexts B_j and O_i using a computation similar to a dot product:

$$D(B_j, O_i) = \frac{\sum_k \{B_j(k) O_i(k)\}}{\sqrt{\sum_k \{B_j(k)\}^2} \sqrt{\sum_k \{Q_i(k)\}^2}}. \tag{1}$$

We can then determine the most similar O_i and B_j for each j:

$$S_j = \max_i \{D(B_j, O_i)\}. \tag{2}$$

To avoid the 180-degree ambiguity of PCA, we take better similarity score D from two possible configurations of the shape contexts pair in computation of S. We will now use $I(j)$ to denote the index i of the shape context O_i which is the most similar shape context to B_j. We now have the most

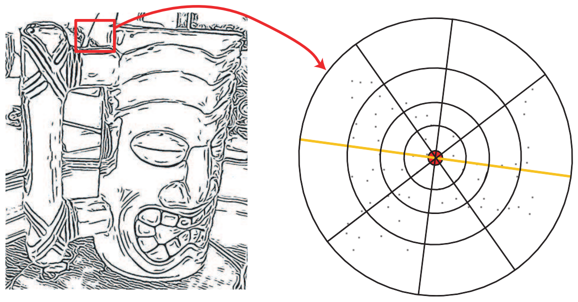


Figure 5: *A rotation-invariant shape context of a local region of a background image.*

locally similar pair $(B_j, O_{I(j)})$ for each j^{th} feature point in the background image.

We now refine the similarity value S_j by considering the similarity between the shape contexts in the neighborhoods of $O_{I(j)}$ and those in the neighborhoods of B_j. Each of these neighborhoods consists of the T numbers of nearest feature points based on the center points, $p_{I(j)}$ and q_j. We can then denote the shape contexts defined in the neighborhoods of $O_{I(j)}$ and B_j as $O_{I(j),t}$ and $B_{j,t}$, $(t = 1, \ldots, T)$, respectively. A new and more refined similarity computation can now be formulated:

$$\hat{S_j} = S_j \left\{ \sum_{t=1}^{T} D(B_{j,t}, O_{I(j),t}) \right\} / T + w\,F\left(B_j, O_{I(j)}\right). \quad (3)$$

The first term allows a higher similarity to the two neighborhoods for a better match between B_j and $O_{I(j)}$. This mitigates the limitation of our shape context, which is defined with respect to each feature point. In effect, the set of shape contexts w.r.t the points in the neighborhood are used as a more exact shape descriptor of the shape of the object or of a local region in the background. The second term in Equation (3) uses the scale difference function F to express the difference in scale between the object image and the target region in the background image:

$$F(B_j, O_{I(j)}) = \frac{\min\left(r(B_j), r(O_{I(j)})\right)}{\max\left(r(B_j), r(O_{I(j)})\right)}, \quad (4)$$

where, $r(B_j)$ and $r(O_{I(j)})$ are the radii of the histogram disks in B_j and $O_{I(j)}$ respectively. Note that the scale difference function F increases as the difference in size between the two radii decreases. This counters the possibility of excessive scaling of the hidden object to fit the target region by penalizing matches between an object and a region whose size are very different. Our implementation used the eight-neighborhood of each feature point and we set the weight w in the second term of Equation (3) to 0.3.

4.3. Finding Hidden Spot and Object Selection

Now, we have to determine where the current object will be hidden using the values of $\hat{S_j}$. Since $\hat{S_j}$'s are defined at discrete feature points in the background image, we need to interpolate them to create a continuous similarity map. We cannot use general spline interpolation, because the low density of the feature points can cause over-fitting, which disrupts the search for a suitable location for a hidden object: an over-fitted interpolation may result in a mapping function which only has peaks at the feature points. To avoid this problem, we use thin-plate spline (TPS) interpolation [Wah90], which minimizes the number of wiggles in the interpolation surface (see Figure 6). The highest point in this smoothed map is taken as the place to hide the object.

We calculate the degree of similarity of all the objects in the database and start by hiding these objects which match best. This ordered candidate list can alternatively be supplied to puzzle creator.

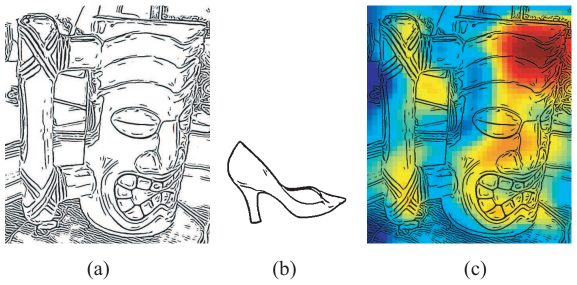


Figure 6: *Similarity map interpolating values of $\hat{S_j}$ using thin-plate spine interpolation: (a) background image; (b) object to be hidden; (c) the similarity map for (a) and (b). Red represents high and blue represents low similarities.*

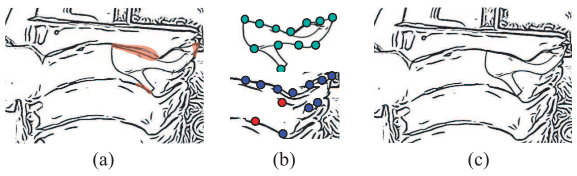


Figure 7: *Affine transformation: (a) a gap existing between the embedded hidden object and the background image (red region); (b) nearest points of feature points are found including some outliers (red points); (c) after applying the affine transformation the object is more closely fitted into the background.*

5. Object Transformation

To determine the final orientation and scale of each object to be hidden, we consider the feature point q_j of the background which is nearest to the hidden place which was determined in the previous section. Let B_j be the shape context

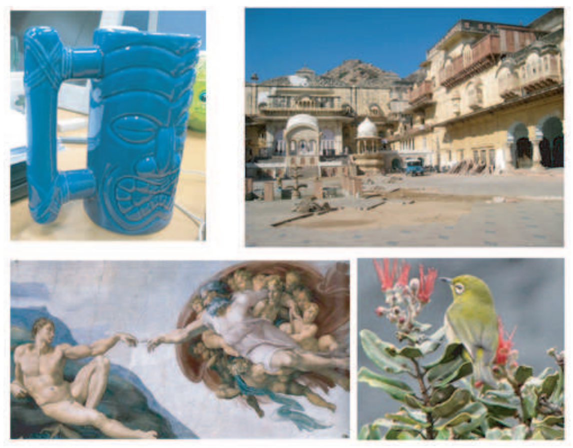


Figure 8: *Experimental background images which are used with generic permission http://creativecommons.org/licenses/by/2.0/deed.en and http://creativecommons.org/licenses/by-nc/2.0/deed.en (The upper-right, lower-left and lower-right images were photographed by Carlton Browne, Feuillu and Eye of einstein, respectively [Fli08].)*

at q_j and let $O_{I(j)}$ be the shape context at the corresponding feature point in the object. Then the hidden object is rotated to make the representative axis of $O_{I(j)}$ coincide with that of B_j. The scale factor required can easily be determined by comparing the radii of the histogram disks of B_j and $O_{I(j)}$. Figure 7(a) illustrates an example of a hidden object embedded in a background image after rotation and scaling. The place where the object is hidden is chosen by interpolation while the rotation and scale are computed for the nearest feature point to that location. This means errors are likely to be present after the transformation, and the object can be embedded less noticeably in the background if a final deformation is performed.

This deformation aims to enhance the similarity of object and background, but it must not destroy the identity of the hidden object. We therefore limit the deformation by using an affine transformation, because the only non-rigid elements that this includes are (uniform and non-uniform) scaling, shearing and reflections. The feature points $p_i(=(p_i^x, p_i^y))$ of the hidden object are transformed into the same number of corresponding feature points $q_i(=(q_i^x, q_i^y))$, in the background image. A 3×2 affine transformation can be obtained by solving

$$A[p_i^x \;\; p_i^y \;\; 1]^T = [q_i^x \;\; q_i^y], \qquad (5)$$

which is a simple linear system.

Some pairs of points (p_i, q_i) may produce an inaccurate affine matrix, and thus we need to disregard these outlier pairs. Liu et al. [LTF*05] introduced a method for computing a stable affine matrix for matching problems using an iterative process. If A^0 is the initial affine matrix, calculated using Equation (5), then at each step we can formulate the probability of a pair being an outlier:

$$P_i^t = \exp\left\{-||A^t[p_i^x \;\; p_i^y \;\; 1]^T - [q_i^x \;\; q_i^y]||^2/(2\delta)\right\}, \qquad (6)$$

where δ is the mean of $||A^t[p_i^x \;\; p_i^y \;\; 1]^T - [q_i^x \;\; q_i^y]||^2$, excluding the outlier pairs computed before step $t-1$. P_i^t represents the probability of a pair (p_i, q_i) being an outlier at the t^{th} iteration. We disregard the outlier pairs which do not satisfy the condition $P_n^t > 1.3\Omega$, where Ω is the mean of P_i^t, excluding the outlier pairs found before step $t-1$ (see [LTF*05]), and then we recalculate the affine matrix A^{t+1}. This process is repeated until no pairs are classified as outliers. Figure 7 shows an example of this process.

Once all the objects have been hidden, we superimpose the lines of objects on the original background image and repeat edge detection using the CLD method. This connects discontinuous edges, producing a more uniform image which enhances the difficulty of the puzzle.

6. Experimental Results

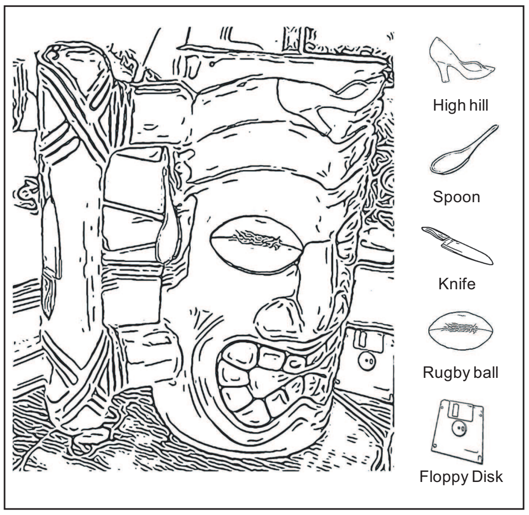
Figure 9, 10, and 11 contain hidden-picture puzzles generated from the background images in Figure 8.

Figure 9(a) hides five objects selected by our system. Figure 9(b) shows another puzzle, a combination of line drawing (obtained by CLD method) with background colors abstracted with the method of Winnemöller et al. [WOG06]. To reduce the color artifacts of the background image with line boundary the hidden objects, we decreased the contrast of the background image slightly. Figure 10 shows another colored hidden-picture puzzle with a more complicated background image.

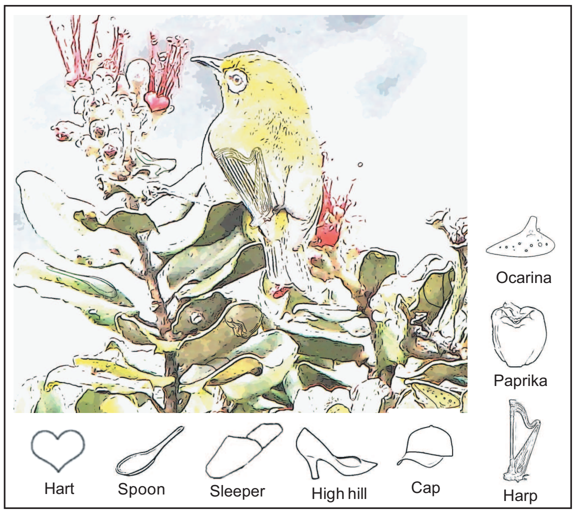
In making Figure 11, we prepared a set of letters to hide and tested them on three candidate background images. This figure shows the one with the highest similarities.

We asked 30 puzzle-solvers with no knowledge of our system to try six puzzles generated by the our system and one puzzle created by Ball (see Figure 1(a)), and to assign scores between zero (for dissatisfactory case) and one (for satisfactory case) to each object after it has been found, expressing their level of satisfaction with the way it was hidden. [†] The results are shown in Table 1, from which we can see that, on average, similarity correlates with the percentage of satisfactory case. It is not surprising that our computer-generated puzzles received slightly lower scores than the one by an experienced puzzle-designer. However, since the solvers' satisfaction increases with the average similarity between the hidden objects and the background, we believe it is possible

[†] All results are available at: http://visualcomputing.yonsei.ac.kr/personal/yoon/hpp/hpp.htm



(a)



(b)

Figure 9: *Automatically generated hidden-picture puzzles: (a) line-drawing style; (b) with a limited number of colors.*

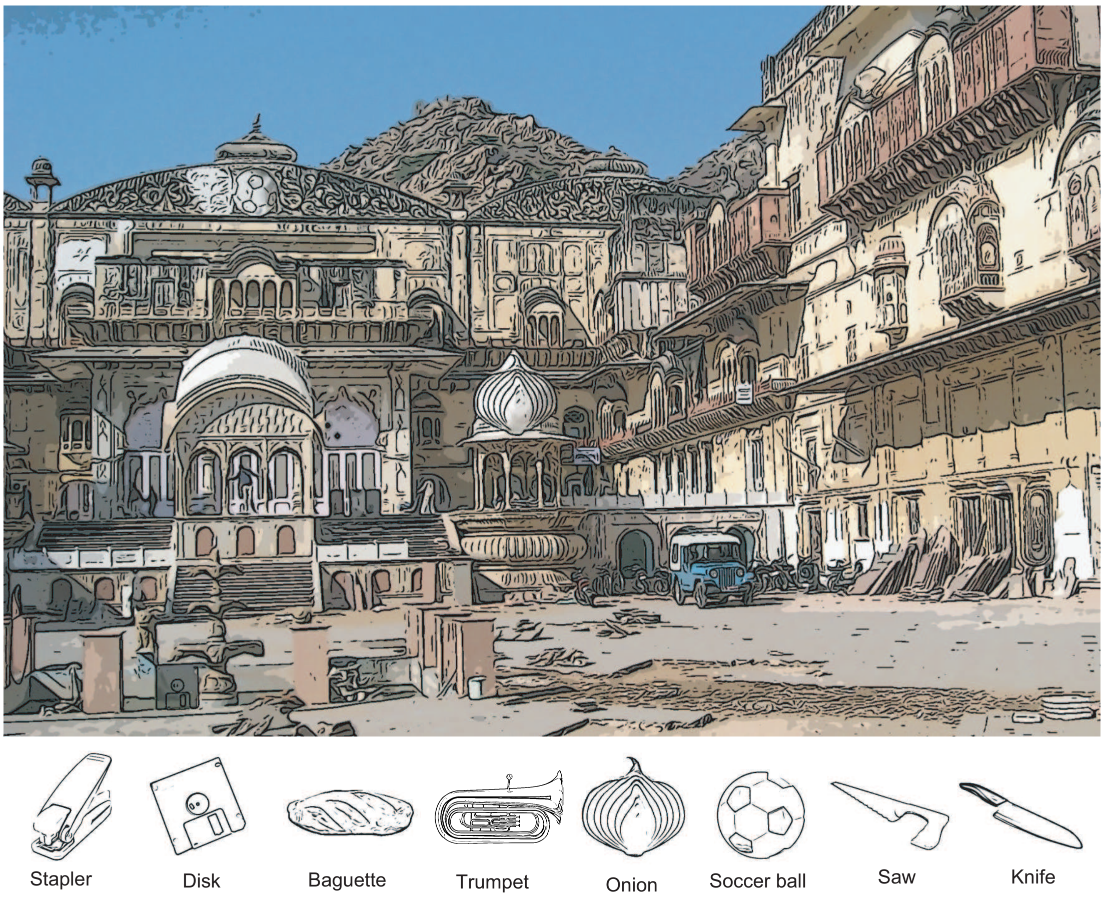


Figure 10: *Hidden picture puzzle based on a complicated background image.*

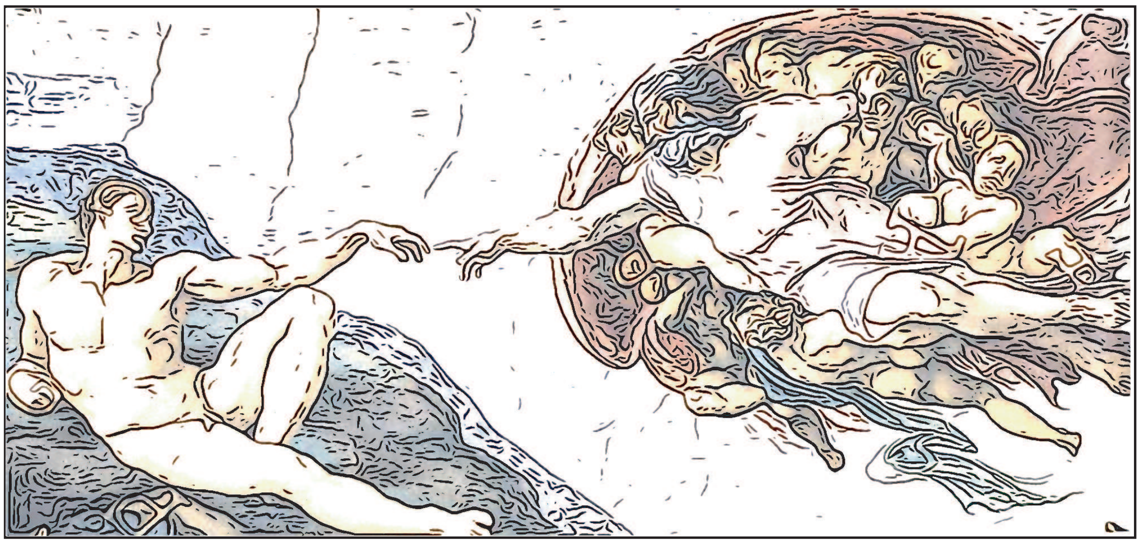


Figure 11: *An automatically generated hidden-picture puzzle: the background is from Michelangelo's Ceiling of the Sistine Chapel, and the hidden objects are the letters , 'G', 'R', 'A', 'P', 'H', 'I', 'C' and 'S'.*

to catch up to or even surpass the quality of a manual puzzle design, with further improvement of our program.

It is interesting to see that the solvers' search time is apparently unrelated to the similarity between an individual hidden object and the place where it is hidden. We suspect that the overall difficulty of a puzzle may also depend on other factors, such as the complexity of the background and the shapes of the hidden objects. Figure 12 confirms that the similarity (between a hidden object and its background structure) is irrelevant to the search time, but does affect the satisfaction level.

Puzzle	# of Feature Pts in Background	Average Similiarity (Maximum 1)	Total Search Time (sec)	Average Satisfaction (%)
Figure 1(a)	-	-	308.4	88.9
Figure 9 (a)	3016	0.774	50.1	76.7
Figure 9 (b)	4091	0.812	111.6	85.9
Figure 10	8923	0.784	445.2	82.1
Figure 11	2984	0.756	60.1	73.6

Table 1: *Puzzle-solvers' evaluation: object similarity against search time and satisfaction.*

The computation time is shown in Table 2. We used Intel Core2 2.14GHz PC with 2GB memory. Constructing the object database does not take long as each object image is small (see Table 2(a)): a hundred objects can be processed in about two minutes. The time for generating similarity map increases in proportion to the number of feature points (Table 2(b)). By careful timing of the code, we learned that a large portion of this computation time is devoted to TPS interpolation. The use of a more efficient interpolation scheme should reduce the total processing time.

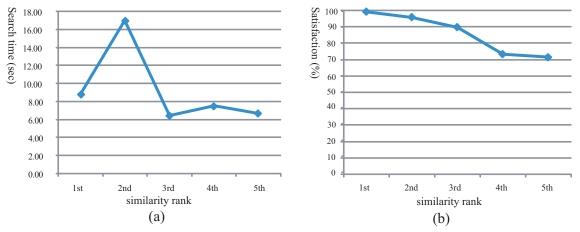


Figure 12: *The relation between (a) similarity and search time, and (b) similarity and puzzle solvers' satisfaction. The data are averages for the hidden objects with the top five similarities in each puzzle.*

	Resolution	Line Drawing	Shape Context
Background	2000 × 1500	7	2
Object	300 × 300	0.9	0.15

(a) Average preprocessing times (sec).

Puzzle	# of Feature Pts in Background	Shape Matching Time per Object	Deformation Time per Object
Figure 11	2984	5.2	< 1
Figure 9 (a)	3016	6.4	< 1
Figure 9 (b)	4091	10.2	< 1
Figure 10	8923	19.2	< 1

(b) Average shape matching and deformation times for one hidden object (sec).

Table 2: *Time required to generate a hidden-picture puzzle.*

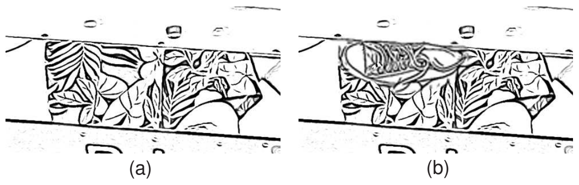


Figure 13: *Undesirable result by our system (a) Background image; (b) Attempt to hide an object of a complex form.*

Satisfaction rank	1st	2nd	3rd	4th	5th
DoLE	0.0663	0.1346	0.2271	0.1985	0.3588

Table 3: *Relationship between the user satisfaction and difference of local entropy (DoLE): the level of satisfaction of hidden objects varies inversely as DoLE value (average of 6 puzzles).*

7. Conclusions

We have introduced an automatic system for generating hidden-picture puzzles. By replacing the skills needed for image stylization and shape matching, our system can effectively support non-expert puzzle creators. It allows arbitrary background images, and can therefore be used in a wide range of application areas such as education, special events, greeting cards, newspapers, and commercials. Our system may also be able to help professional artists in planning hand-drawn puzzles.

Future research can bring improvements to our system. Our generator basically compares the distributions of the sampled feature points between the target object and the background image to find the right place to hide, and sometimes this is not enough, especially when the target object has a complex structure that may not be properly described by feature points only. In this case, the target object may be hidden in an undesirable location (see Figure 13). We experimented with the difference of local entropy [ZF03] between the original background image and the puzzle. As expected, the region of unsatisfactory case has a relatively large difference value (see Table 3). These values of entropy differences can be incorporated into our system to filter out undesirable shape matching. Additionally more robust algorithm is required that involves the curves connecting the feature points for a more rigorous shape matching.

Another limitation is the number of objects we can put in the database. While adding more objects could improve the quality of puzzle, it will also increase the search time. One way to reduce the search time could be the hierarchical structuring of objects in the database [LHA*07].

We may also be able to further improve our system and make more challenging puzzles by incorporating the mechanism of human perception.

References

[Bal08] BALL L.: Liz's hidden picture puzzles, 2008. http://www.hiddenpicturepuzzles.com.

[BBM05] BERG A. C., BERG T. L., MALIK J.: Shape matching and object recognition using low distortion correspondences. In *Proc. CVPR* (2005), pp. 26–33.

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *IEEE Trans. Pat. Anal. Mach. Int. 24*, 4 (2002), 509–522.

[CRH05] COLLOMOSSE J. P., ROWNTREE D., HALL P. M.: Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Trans. Visualization and Computer Graphics 11*, 5 (2005), 540–549.

[CTCG95] COOTES T. F., TAYLOR C. J., COOPER D. H., GRAHAM J.: Active shape models: their training and application. *Computer Vision and Image Understanding 61*, 1 (1995), 38–59.

[DOM*01] DURAND F., OSTROMOUKHOV V., MILLER M., DURANLEAU F., DORSEY J.: Decoupling strokes and high-level attributes for interactive traditional drawing. In *Proc. Eurographics Workshop on Rendering* (2001), pp. 71–82.

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *Proc. ACM SIGGRAPH* (2002), pp. 769–776.

[Fli08] FLICKR: Share your photos. watch the world., 2008. http://flickr.com.

[GCS02] GOOCH B., COOMBE G., SHIRLEY P.: Artistic vision: painterly rendering using computer vision techniques. In *Proc. NPAR* (2002), pp. 83–90.

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proc. NPAR* (2004), pp. 113–120.

[Her01] HERTZMANN A.: Paint by relaxation. In *Computer Graphics International* (2001), pp. 47–54.

[HKR93] HUTTENLOCHER D. P., KLANDERMAN G. A., RUCKLIDGE W. A.: Comparing images using the hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 9 (1993), 850–863.

[HS98] HARRIS C., STEPHENS. M.: A combined corner and edge detector. In *Alvey Vision Conference* (1998), pp. 147–152.

[KLC07] KANG H., LEE S., CHUI C.: Coherent line drawing. In *Proc. NPAR* (2007), pp. 43–50.

[KP02] KIM J., PELLACINI F.: Jigsaw image mosaics. In *Proc. ACM SIGGRAPH* (2002), pp. 657–664.

[LHA*07] LALONDE J.-F., HOIEM D., A.EFROS A., ROTHER C., WINN J., CRIMINISI A.: Photo clip art. In *Proc. ACM SIGGRAPH* (2007), p. 3.

[LLE00] LATECKI L. J., LAKÄMPER R., ECKHARDT U.: Shape descriptors for non-rigid shapes with a single closed contour. In *Proc. CVPR* (2000), pp. 424–429.

[Low03] LOWE D.: Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision* (2003), vol. 20, pp. 91–110.

[LTF*05] LIU C., TORRALBA A., FREEMAN W. T., DURAND F., ADELSON E. H.: Motion magnification. In *Proc. ACM SIGGRAPH* (2005), pp. 519–526.

[Mal99] MALCHIODI C. A.: *Medical Art Therapy With Children*. Jessica Kingsley Publishers, 1999.

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. In *Proc. ACM SIGGRAPH* (1997), pp. 401–406.

[VH01] VELTKAMP R. C., HAGEDOORN M.: State of the art in shape matching. *Principles of visual information retrieval* (2001), 87–119.

[VJP97] VETTER T., JONES M. J., POGGIO T.: *A Bootstrapping Algorithm for Learning Linear Models of Object Classes*. Tech. Rep. AIM-1600, 1997.

[Wah90] WAHBA G.: Spline models for observational data. *CBMS-NSF Regional Conference Series in Applied Mathematics 59* (1990).

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. In *Proc. ACM SIGGRAPH* (2006), pp. 1221–1226.

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. In *Proceedings of SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (2004), pp. 574–583.

[XK07a] XU J., KAPLAN C. S.: Calligraphic packing. In *Proc. Graphics Interface* (2007), pp. 43–50.

[XK07b] XU J., KAPLAN C. S.: Image-guided maze construction. In *Proc. ACM SIGGRAPH* (2007), p. 29.

[XKM07] XU J., KAPLAN C. S., MI X.: Computer-generated papercutting. In *Proc. Pacific Graphics* (2007), pp. 343–350.

[ZF03] ZITOVÄ B., FLUSSER J.: Image registration methods: a survey. *Image Vision Comput. 21*, 11 (2003), 977–1000.