Calligraphic Packing

Jie Xu Craig S. Kaplan*

Computer Graphics Lab
David R. Cheriton School of Computer Science
University of Waterloo

Abstract

There are many algorithms in non-photorealistic rendering for representing an image as a composition of small objects. In this paper, we focus on the specific case where the objects to be assembled into a composition are letters rather than images or abstract geometric forms. We develop a solution to the "calligraphic packing" problem based on dividing up a target region into pieces and warping a letter into each piece. We define an energy function that chooses a warp that best represents the original letter. We discuss variations in rendering style and show results produced by our system.

CR Categories: I.3.m [Computing Methodologies]: Computer Graphics—Miscellaneous J.5 [Computer Applications]: Arts and Humanities

Keywords: Tilings, Calligraphy, Mosaics, Non-Photorealistic Rendering, Packing

1 Introduction

Artists and art lovers have always been captivated by the interplay between a whole and its parts. When a complete image is seen as composed of many small cooperating elements, we can appreciate not only the image as a whole, but the ingenuity of its conception and realization. In a Roman mosaic, for example, small squares of coloured glass conspire to form a detailed scene.

Especially fascinating is the case where the parts are themselves recognizable objects. The viewer is then caught in a dynamic tension between attending to the image and to the elements that make it up. A famous example that has inspired many researchers in computer graphics are Arcimboldo's sixteenth century portraits [16]. His subjects are assembled from small objects such as flowers, vegetables, fruits, and animals. Halsman's famous *In Voluptate Mors* is a portrait of Dalí featuring a skull assembled from nudes. In Escher's tessellations, the whole is simply the entire plane, but we are fascinated by the way lifelike characters occupy it without any gaps. Escher was also interested in the narrative possibilities that arise in having multiple different objects interact in an image.

A special case of this artform can be found in what we call "representational calligraphy". Here, a collection of letters or symbols cooperate to compose an image that is also a calligraphic inscription. Islamic calligraphers are certainly the masters of this style. A stunning recent example by Sudanese-born artist Hassan Musa is shown in Figure 2(a). Today, representational calligraphy is frequently used in graphic design applications such as advertising, product design, and corporate logos. The example of Figure 2(b) was part of an advertising campaign for Veja, a Brazilian news magazine.

*e-mail: {jiexu,csk}@cgl.uwaterloo.ca

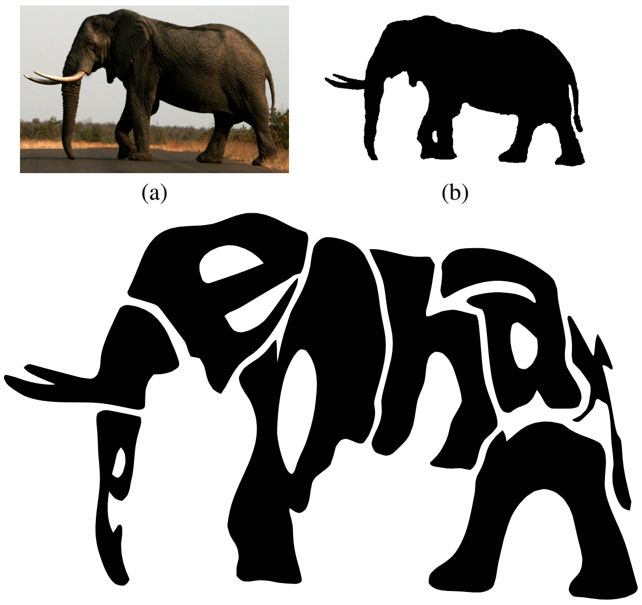


Figure 1: A calligraphic mosaic created by our system. An elephant is extracted from a photograph (a) and converted into a container region (b). The word "elephant" is then packed into the container.

In representational calligraphy the letters are frequently distorted, and the inscription may be difficult or even impossible to read. In some examples, it is possible to recognize the letters only if one already knows (from the context) what they are. Note that there is no harm in deforming letters this way. Legibility and consistency are the goals of *typography*; both may be suppressed for artistic purposes in calligraphy. The resolution of the resulting visual puzzle can be a rewarding aesthetic experience.

Within non-photorealistic rendering (NPR) there has been a great deal of work in the area of "NPR Packing", the general problem of depicting an image by automatically arranging a collection of small pictorial elements. But to our knowledge, no research has been done on the specific problem of packing letterforms. Inspired by the examples above, we therefore pose a representational calligraphy problem for computer graphics:

Problem (Calligraphic Packing): Given a region of the plane (the "container") and a sequence of letters $L = \{l_1, \ldots, l_n\}$, construct a non-overlapping arrangement of deformed glyphs $\{g_1, \ldots, g_n\}$ in the interior of the container so that

1. The glyphs fill the container as much as possible;
2. Individual glyphs are recognizable as the corresponding letters; and
3. The order of the letters is suggested by the arrangement of the glyphs in the packing.

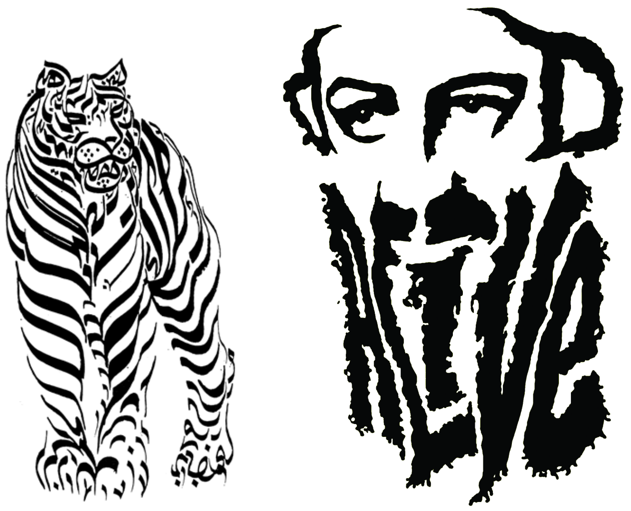


Figure 2: Two examples of representational calligraphy. The tiger on the left, by Sudanese-born artist Hassan Musa, is made up of elongated Arabic letters. The portrait on the right is from an advertisement for the magazine Veja.

Note that we make a distinction between a letter (an abstraction) and a glyph (its representation). A single letter may be equally well represented by any of a number of glyphs, such as lowercase and uppercase versions.

Although calligraphic packing is related to other forms of NPR packing, there are important differences that make this problem unique. First, we don't have a database of available objects, and the freedom to pack copies of objects from the database at will. We are given a specific sequence of symbols, and must include each exactly once. The relative positions and orientations of the letters are important too, as they affect overall readability. Finally, we expect to apply a significant amount of deformation to the glyphs in order to pack them. Unlike most NPR packing applications, our glyphs can undergo a great deal of deformation compared to ordinary images. The glyphs are not intended to be *representations*, merely *signs*. As long as the original letters can be recovered, the incidental shapes of the glyphs are unimportant.

In this paper, we provide a solution to the calligraphic packing problem. We automate the process of warping glyphs to fill the container, but leave the question of natural ordering partly in the hands of the user. Our system uses a clustering algorithm (Section 3.2) to subdivide the container into subregions. We then warp the glyphs into the subregions (Section 3.3) in a way that minimizes distortion (Section 3.4). We also include some variations in rendering style (Section 4). An example created by our system is shown in Figure 1.

2 Related work

Many researchers in non-photorealistic rendering have studied some variation of NPR Packing. As far as we know, no previous work has specifically addressed the packing of letterforms.

One primary thread of NPR packing techniques has focused on the use of Lloyd's method to relax an initial distribution of objects into a final, evenly-spaced configuration. Hausner [7] simulated the appearance of Roman mosaics. He applied Lloyd's method to Voronoi diagrams generated via the Manhattan metric, giving an arrangement of oriented rectangular mosaic tiles. Hiller et al. [9] improved and generalized this result by using area Voronoi diagrams, allowing more complex objects to be packed together. Secord [18] used a weighted version of Lloyd's method as a form of importance sampling to create stippled depictions of images. But for arbitrary tiling shapes like the letters, these methods cannot get tiles that resemble these shapes.

Recently, Dalal et al. [4] achieved excellent packing results by modifying the relaxation step in Lloyd's method. Instead of moving the object to the centroid of its Voronoi region, they define a metric based on image correlation that minimizes variation in the space around the objects (the "grout" in the mosaic). All these packing techniques focus on distributing a large number of small elements without deformation. We wish to pack a small, fixed set of deformable shapes.

Kim and Pellacini [13] presented a more generic framework for packing shapes chosen from a database of candidates into a container. Their algorithm attempts to minimize an energy function that trades off between various measures of the packing's quality. Their technique deforms objects slightly to even out the irregular boundaries between them. We would like to support an even greater amount of deformation and provide some mechanism for controlling the flow of the letters in order to maintain readability.

Kaplan and Salesin [12] examined the problem of Escherization, in which a given shape must be converted into a tiling of the plane. Their algorithm seeks a tileable shape that resembles the original as closely as possible, thereby minimizing the amount of deformation that must be applied.

In the artistic screening method of Ostromoukhov and Hersch [17], shapes such as letterforms were continuously deformed via interpolation from a small set representing different tones. Grids of these shapes were then used for halftoning by varying the interpolation level spatially. Surazhsky and Elber [20; 21] presented a method to arrange text along free-form parametric curves. They rendered their text with varying tone along projected parametric curves on surfaces in order to produce comprehensible text-based rendering of 3D shapes. Compared to these techniques, our deformations are more freeform. Also, our results are not halftoned representations; we pack a container derived from a bi-level image.

3 Approach

We have created a system that produces calligraphic mosaics. Given an image and a sequence of letters, we convert the image into a black and white representation of an object (Section 3.1), subdivide it into regions (Section 3.2), and warp the individual letters into these regions (Sections 3.3 and 3.4).

3.1 Container extraction

The first step in our system is to define the container region into which we will pack the letter sequence. When drawn as a black shape on a white background, the container should be a clear depiction of an object.

In our system, the user first extracts a desired foreground object from an image using the Lazy Snapping method of Li et al. [15]. We then apply a Gaussian blur to the object to smooth out high frequency details. Finally, we convert the object to greyscale and threshold it to produce a bi-level raster image of the container region. The user can control the threshold value to select an appropriate container region. This procedure is illustrated in Figure 3.

3.2 Subdivision

The pixels of the container region must now be partitioned into n subregions $R_1, \ldots, R_n$, where each R_i will be assigned to l_i from the

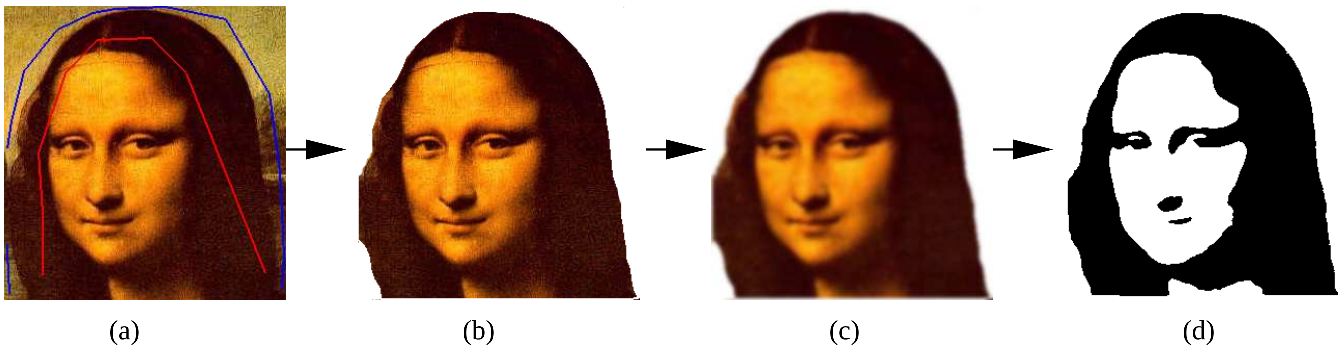


Figure 3: The extraction of a container region from an image. The user marks the input image in (a) with foreground (red) and background (blue) curves. Lazy Snapping extracts the foreground object in (b). The blurred image in (c) is thresholded by brightness to obtain the bi-level container in (d).

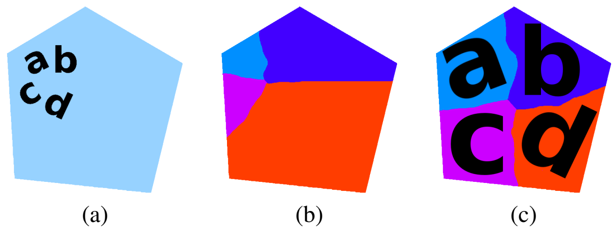


Figure 4: The user provides initial positions and orientations for a set of letters in (a). The initial clustering result is shown in (b). In (c), after iterative relaxation, we obtain an optimized clustering result.

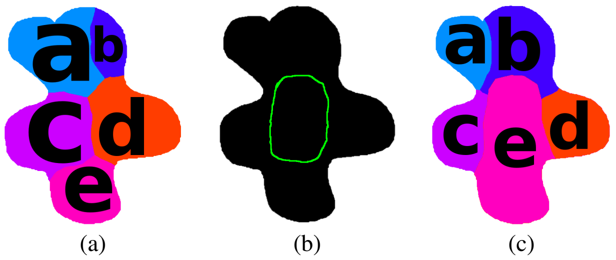


Figure 5: The forced clustering of pixels. The clustering algorithm produced the result in (a). The user drew the green curve in (b). The pixels inside the curve are forced to belong to a single cluster in (c).

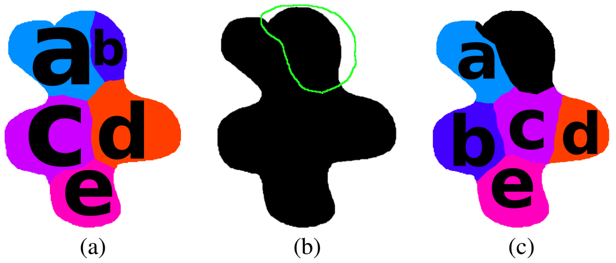


Figure 6: The forced exclusion of pixels from the subdivision process. The user sketched the green curve in (b). Its contents do not participate in the subdivision in (c).

original letter sequence. This step can be seen as a data clustering problem, where pixels form into clusters around letters.

As with other NPR packing techniques, we use an iterative algorithm to refine an initial letter placement into an evenly distributed set of subregions. To begin, the user interactively creates a starting arrangement of n letters inside the container region, as shown in Figure 4(a). We then run a level-set algorithm to grow subregions associated with the letters [19]. (The use of a level-set algorithm helps ensure that letters are contained in their subregions, and that the subregions are connected.) We initialize n clusters to the rasterized outlines of the letters. Then, while there exist unclaimed pixels, we iterate over the clusters. Each cluster claims the pixels adjacent to its boundary. The subregion boundaries will grow at a roughly constant rate until they encounter one another and consume the container. The resulting subdivision may be uneven, as in Figure 4(b). We refine it by moving each letter to the centroid of its cluster, in the style of the modified Lloyd's method of Hiller et al.[9]. Each letter is scaled so that it fits more snugly into its cluster. The subdivision process then repeats until the user is satisfied with the locations and orientations of the letters. Figure 4(c) shows an example of a final set of subregions produced by this process.

We provide the user with two additional interactive tools to modify the behaviour of the subdivision step. First, they can sketch closed curves containing pixels that must all belong to the same subregion. A demonstration is given in Figure 5. This tool can be used to force an area of the container to belong to a single letter when the subdivision process is trying to pull it apart. It was used in Figure 12 to ensure that the mouth would be depicted by a single subregion containing the letter O. Second, the user can sketch a closed curve that excludes a portion of the container from the subdivision, as shown in Figure 6. In Figure 16(b), the propellers of the airplane were simply drawn as black shapes rather than as awkward protrusions on the letter D.

At this point, the subdivision process has produced a partition of the container region into sets of pixels. We must now convert these subregions into paths that can serve as geometric targets for image warping. We smooth the boundaries of the subregions by applying a few iterations of the Open and Close morphological operations. We finish by applying Open three more times to shrink the subregion boundaries and create space between them. A library such as Autotrace [23] can then convert the rasterized subregions into vector paths. If desired, the user can edit the paths manually before proceeding with the warping step. A set of subregions is shown in Figure 11(b).

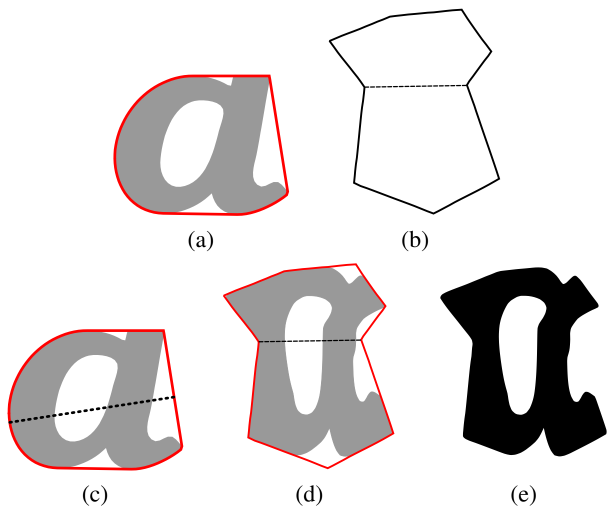


Figure 7: A visualization of the warping process. A source letter and its convex hull are shown in (a). A container subregion is shown in (b), partitioned into convex pieces. The corresponding subdivision is applied to the convex hull in (c), and the letter is warped piece-by-piece into the convex partition in (d). The finished warped letter is shown in (e).

3.3 Warping

Now that we have the container subregions, we must warp the corresponding glyphs into them. Warping and morphing are well studied problems in computer graphics [5]. Unfortunately, in most cases the user must provide an explicit correspondence between the source and target shapes. We would like the computer to derive this correspondence automatically based on the quality of the warp it produces. We provide a geometric warping technique that incorporates an energy function based on shape matching to measure the success of the warp. We can then try a large set of possible correspondences and choose the one that minimizes the energy function. We discuss the warp in this subsection, and the energy function in the next one.

Let us denote by C_i the convex hull of glyph g_i, which will be warped into subregion R_i. An example is given in Figure 7(a). We will define a warp from C_i to R_i, and apply it to the glyph. To begin, we place the same number of sample points evenly around both C_i and R_i (we have found that 100 points is an adequate trade-off between efficiency and accuracy). Any given assignment of a point on C_i to a point on R_i induces a correspondence between the two shapes.

Hormann and Floater [10] present a technique for warping between two arbitrary simple polygons, based on a generalized notion of barycentric coordinates. We find that their technique performs well when the source and target polygons are convex, but produces unacceptable distortion in general. While C_i is convex by definition, R_i need not be. Therefore, we partition R_i into convex pieces using the approximation algorithm of Greene [6]. Figure 7(b) shows an example of a partitioned subregion. Given a correspondence between the sample points on C_i and R_i, we can then map this partition back through the correspondence and create a convex partition of C_i, shown in Figure 7(c). Finally, we apply the geometric warp of Hormann and Floater inside each convex piece to obtain a warp of the entire glyph. The outlines of the glyph are approximated by piecewise-linear paths with m vertices $\{v_1, \ldots, v_m\}$, and the glyph is warped by computing new positions $\{w_1, \ldots, w_m\}$ for each of the vertices in those paths. A finished example is given in Figure 7(e).

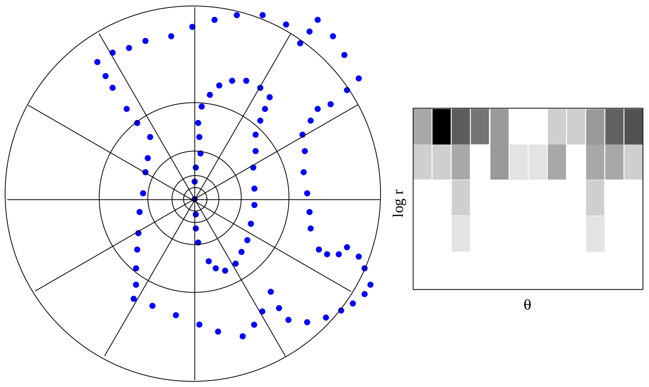


Figure 8: A visualization of the shape context for a single reference point on a warped letter "a". The plane is divided into log-polar regions on the left, and the sample points are collected into histogram bins on the right.

3.4 Shape Matching

As was mentioned in the previous section, if we fix a sample point on C_i and assign it in turn to each sample point on R_i, we get a sequence of correspondences between the two shapes. How do we decide which one is the best? Here, we need a measurement that evaluates the quality of the warp resulting from a given correspondence. Our measurement is based primarily on geometric similarity between the warped glyph and the original. We also penalize warps that rotate the glyph too much or fail to fill the container subregion.

The measurement of similarity between two shapes is an important and active area of research in computer vision. We have found that the "shape context" method of Belongie et al. [2] is a good fit to our needs. In this method, the shapes to be compared can have arbitrary topology (unlike, for example, the method of Arkin et al. [1]). They need only have the same number of sample points, which ours do by construction.

A shape context for a single reference point is a log-polar histogram of the locations of the other points in the shape. The histogram is discretized by logarithmic distance from the reference point and by angle. An example is given in Figure 8.

Belongie et al. explain how to compute $\delta(v_i, w_j)$, the distance between the histograms associated with reference points v_i and w_j on the original and warped copies of the glyph. We can then define a measurement of geometric similarity as $\Delta_g = \frac{1}{m}\sum_{i=1}^{m} \delta(v_i, w_i)$. Note that the original use of shape contexts minimized over all pairs of reference points; here, we exploit the known correspondence between the v_i and w_i.

In order to preserve readability, it is important to avoid rotating glyphs too much. In the worst case, a rotated letter may turn into a different letter entirely, as an "m" becomes a "w" under a half-turn. Therefore, we define a penalty Δ_o based on the effect of the warp on the glyph's orientation. We use a least-squares estimate [22] of the rigid motion between the v_i and w_i to compute the change θ in orientation induced by the warp. We can then define $\Delta_o = \theta/\pi$.

We would also like to force warped glyphs to occupy as much of their container subregions as possible. We introduce an additional penalty based on area coverage: $\Delta_a = 1 - A_w/A_r$, where A_w is the area of the warped glyph and A_r is the area of its container subregion.

Finally, we define the overall quality of a matching as $\Delta = \alpha\Delta_g + \beta\Delta_o + \gamma\Delta_a$. The values α, β and γ are weights chosen to trade off between the relative importance of these three terms. In practice, we use $\alpha = 0.6, \beta = 0.2$ and $\gamma = 0.2$. Figure 9 shows some

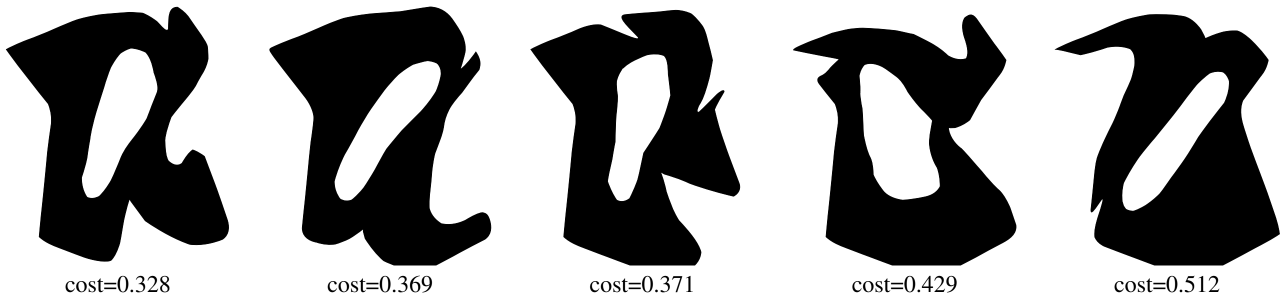


Figure 9: The costs of warping letters. Smaller values indicate a closer match to the original letter of Figure 7.

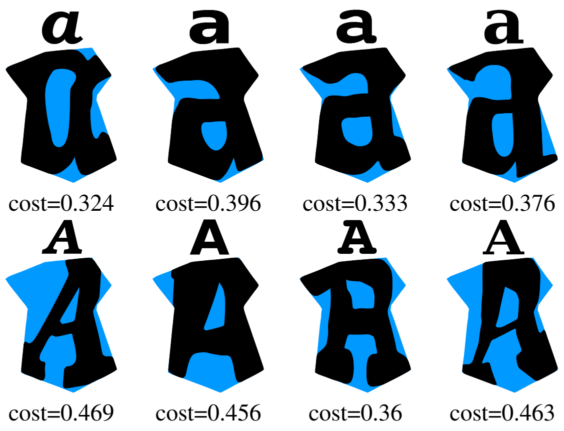


Figure 10: The best-fit costs of lowercase and uppercase versions of the letter "a" for four different typefaces. The original letters are shown on the top, and are packed into the blue container subregions.

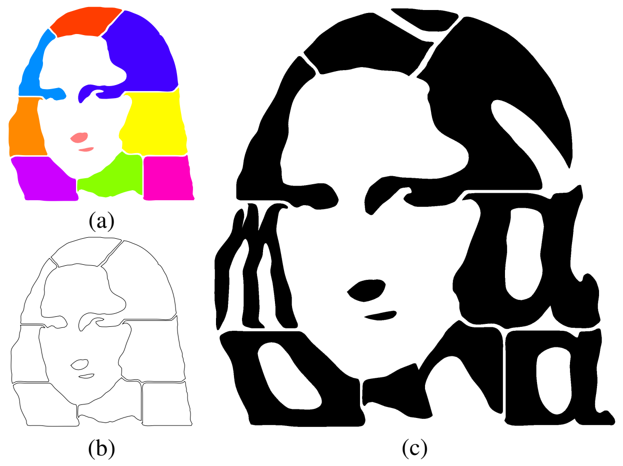


Figure 11: A calligraphic mosaic made from the painting and words "Mona Lisa". After we subdivide the image (a) and extract the outlines (b), our system will produce the finished result in (c). The details of the nose and mouth were excluded from the packing process, and taken directly from the container.

warped letters and their associated costs. We compute Δ for every possible correspondence between C_i and R_i, and choose the warp that produces the lowest cost.

We extend this cost function in a simple but important way by trying multiple glyphs for each letter. We iterate over a few different typefaces, and try both lowercase and uppercase versions of letters. Notice, for example, the use of mixed case in Figure 17(b). The examples in this paper were constructed from the four typefaces *URW Bookman*, *Bitstream Vera Sans*, *Courier*, and *DejaVu Serif*. The additional variety of multiple letterforms helps to ensure a better match.

A finished example of our calligraphic packing technique, based on the Mona Lisa, is shown in Figure 11.

4 Rendering

We usually render the warped letters as solid black shapes. We have also experimented with a few additional rendering styles in order to increase the aesthetic appeal of our results.

Inspired by the rough outlines in Figure 2(b), we allow boundaries of letters to be perturbed geometrically before rendering. This style is useful for capturing the look of rough or irregular objects, such as Lincoln's facial hair in Figure 12.

We also allow the interior of a letter to be filled with strokes instead of a solid colour. The user is able to draw a few sample curves to define the principle stroke direction in a letter. We interpolate from points sampled on those curves using radial basis functions to infer a vector field, in the manner of Hays and Essa [8]. The streamline placement algorithm of Jobard and Lefer [11] draws long, curved strokes that follow the vector field. These strokes give us an opportunity to experiment with texture and tone in our designs. For example, Figure 13 shows streamlines used to give the look of a reflection in water.

5 Implementation and Results

Our system is implemented in C++, and generates Postscript output. We use the CGAL library [3] to compute the convex hull and convex partition. We implemented a prototype user interface in Gtkmm that lets the user load in an image, create the binary container image, fill it with letters, and modify the rendering parameters. The automated mosaic process usually takes from a couple of seconds to a couple of minutes to complete on a standard PC, depending on how many letters are used.

As mentioned throughout the paper, we provide a wide range of interactive tools that permit the user to intervene in the automatic construction of a mosaic. They can edit the curves representing the subregions and the shapes of the warped letters. For example, we added some detail to Chaplin's eyes in Figure 17(b).

Of course, our system is not restricted to the Roman alphabet; it can process more complex symbols. Figure 14(a) shows a result in which three copies of the Chinese word "bird" (written in the traditional zhuànwén, or "seal script") are used to fill the outline of

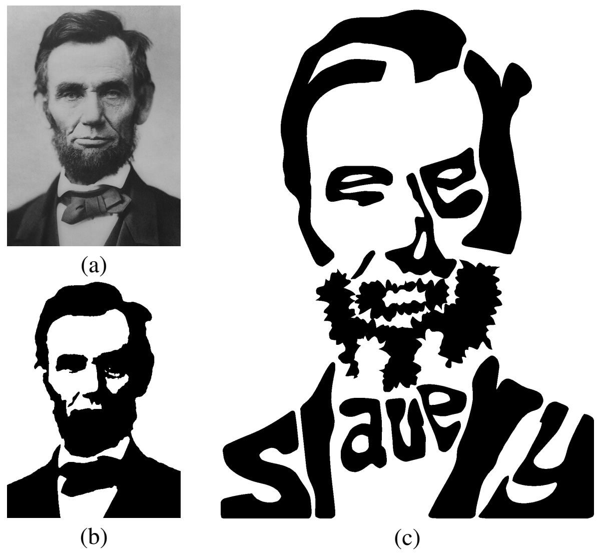


Figure 12: Abraham Lincoln (a) was faced with the conflict between "freedom" and "slavery". The container image is shown in (b). In the result (c), we added noise to the outlines of the letters O and M to give the appearance of Lincoln's facial hair.

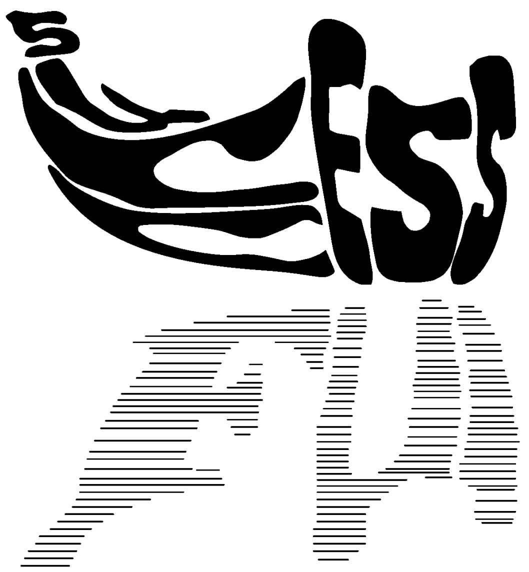


Figure 13: The word "successful" made into a mosaic of a boat and its reflection.

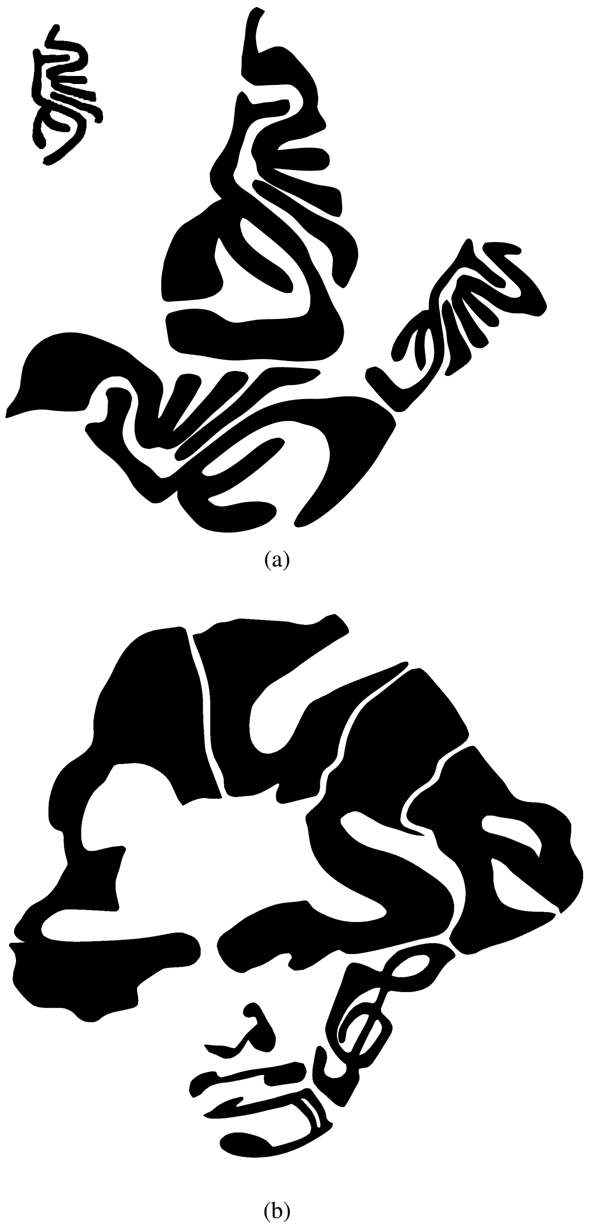


Figure 14: Three copies of the Chinese character for "bird" are used to fill a bird in (a). The source character is shown on the upper left. "Muse" and some musical notation are used to create a portrait of Ludwig van Beethoven in (b).

a bird. Figure 14(b) includes some musical symbols in a portrait of Beethoven.

Some additional results are shown in Figures 15, 16 and 17.

6 Conclusion

In this paper, we introduced the calligraphic packing problem and proposed a solution. Our system uses a level-set approach to subdivide a container region into subregions, and a best-fit warping algorithm to map letters into subregions. We also developed some variations in rendering style that provide the user with additional aesthetic opportunities.

Like other problems in non-photorealistic rendering, calligraphic packing is a challenging topic because a successful solution relies on facts from both mathematics and human perception. Calligraphy is an especially interesting medium in this regard, because it relies on the cognitive and perceptual mechanisms of reading, about which far too little is understood.

We note two important directions in which the technique in this paper can be improved. First, we would like to solve the third requirement of our calligraphic packing problem, namely that the arrangement of glyphs respect the order of original sequence of letters. We would like to find a way to distribute letters automatically so that the eye is naturally drawn across them in the order in which

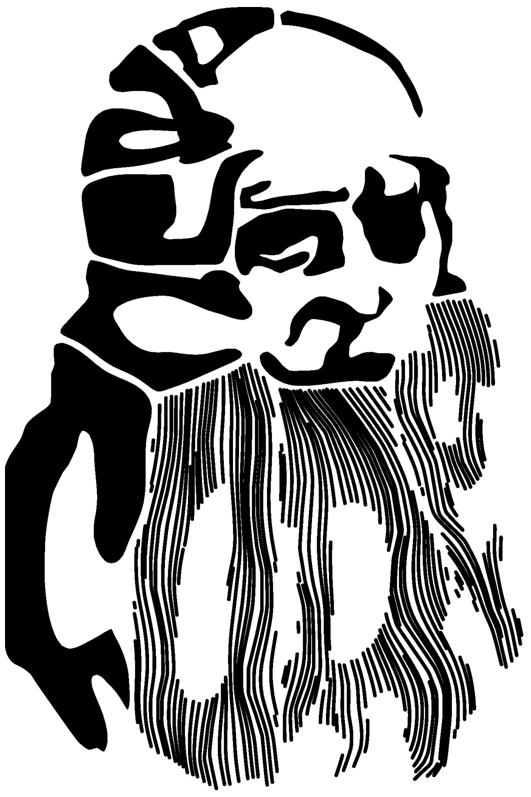


Figure 15: The "Da Vinci Code" packed into a portrait of Leonardo.

they are intended to be read. We would have to extend our energy function to account not only for the quality of each individual warp, but for the layout of the glyphs as a whole.

Second, we would like to improve the legibility and fit of the rendered letters. We cannot simply reduce the convex hull to the boundary of the glyph, because some of the negative space adjacent to the letter is essential in identifying it (consider, for example, the spaces separating the arms of an uppercase "E"). However, the convex hull also contains incidental features such as serifs that reduce the quality of the packing. It may be necessary to move to a spine-based model of letter construction, where we pack zero-thickness drawings of letters and expand outward from them to fill the container. A parameterized model of letterforms like that of METAFONT [14] could be used to control the shapes of letters during this process.

In the long run, we are interested in ways that the computer can be used as a tool for calligraphic design. Computers are long established as a powerful technology for typography, but more research needs to be done to achieve the same level of power and flexibility with letterforms that non-photorealistic rendering grants us with images.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback. We are also very grateful to Mark ter Laan for the elephant photograph in Figure 1, to Hassan Musa for permission to print the tiger in Figure 2(a), to Bubbles Incorporated SA for permission to use Chaplin image in Figure 17(b), and to AlmapBBDO for permission to use the image in Figure 2(b).

References

[1] Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):209–216, 1991.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, 24(4):509–522, 2002.

[3] CGAL. CGAL:computational geometry algorithms library. `http://www.cgal.org`, 2006.

[4] Ketan Dalal, Allison Klein, Yunjun Liu, and Kaleigh Smith. A spectral approach to npr packing. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 71–78, 2006.

[5] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

[6] D. Greene. The decomposition of polygons into convex parts. In *In Advances in Computing Research*, pages 235–259. JAI Press, 1983.

[7] Alejo Hausner. Simulating decorative mosaics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 573–580, New York, NY, USA, 2001. ACM Press.

[8] James Hays and Irfan Essa. Image and video based painterly animation. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120, New York, NY, USA, 2004. ACM Press.

[9] Stefan Hiller, Heino Hellwig, and Oliver Deussen. Beyond stippling - methods for distributing objects on the plane. *Computer Graphics Forum*, 22(3):515–522, 2003.

[10] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, 2006.

[11] Bruno Jobard and Wilfrid Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing '97. Proceedings of the Eurographics Workshop in Boulogne-sur-Mer, France*, pages 43–56, Wien, New York, 1997. Springer Verlag.

[12] Craig S. Kaplan and David H. Salesin. Escherization. In Kurt Akeley, editor, *Siggraph 2000*, pages 499–510. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[13] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaic. In *SIGGRAPH 2002 Conference Proceedings*, pages 657–664, 2002.

[14] Donald E. Knuth. *The METAFONT book*. Addison-Wesley, 1986.

[15] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[16] O Mataev and H Mataev. Olga's gallery. giuseppe Arcimboldo. `http://www.abcgallery.com/A/arcimboldo/arcimboldo.html`, 2006.

[17] Victor Ostromoukhov and Roger D. Hersch. Artistic screening. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 219–228, New York, NY, USA, 1995. ACM Press.

[18] Adrian Secord. Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43. ACM Press, 2002.

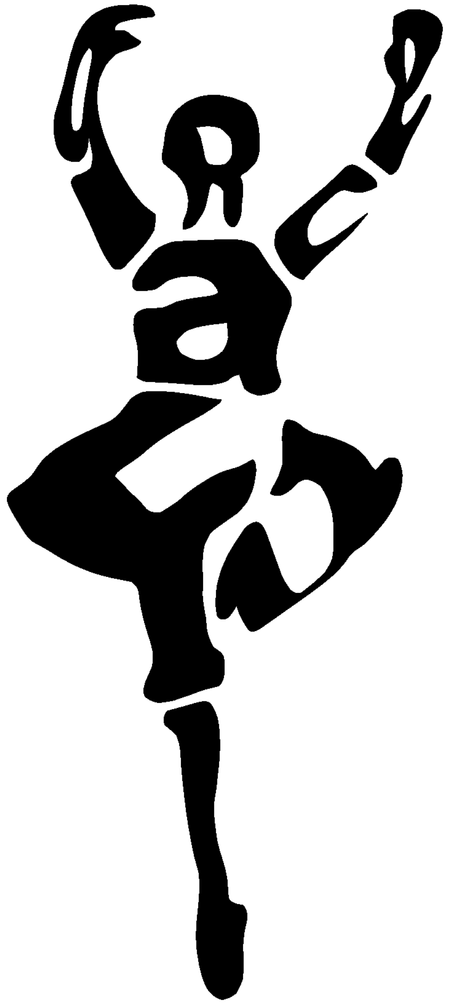
[19] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[20] Tatiana Surazhsky and Gershon Elber. Arbitrary precise orientation specification for layout of text. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 80, Washington, DC, USA, 2000. IEEE Computer Society.

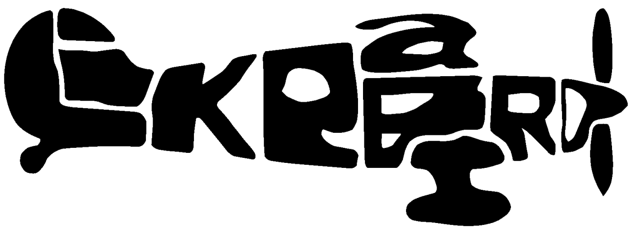
[21] Tatiana Surazhsky and Gershon Elber. Artistic surface rendering using layout of text. *Computer Graphics Forum*, 21(2):99–110, 2002.

[22] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.

[23] Martin Weber. Autotrace - converts bitmap to vector graphics. `http://autotrace.sourceforge.net/`, 2006.

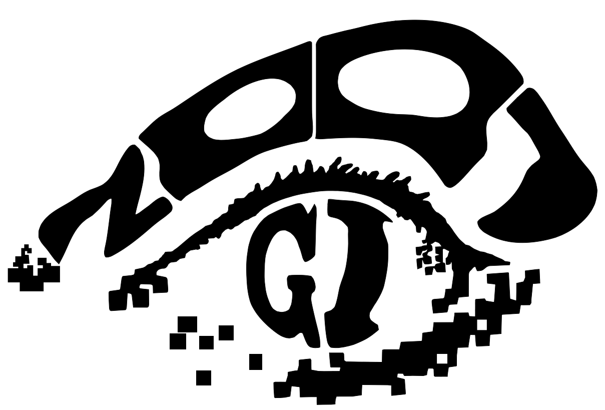


(a)

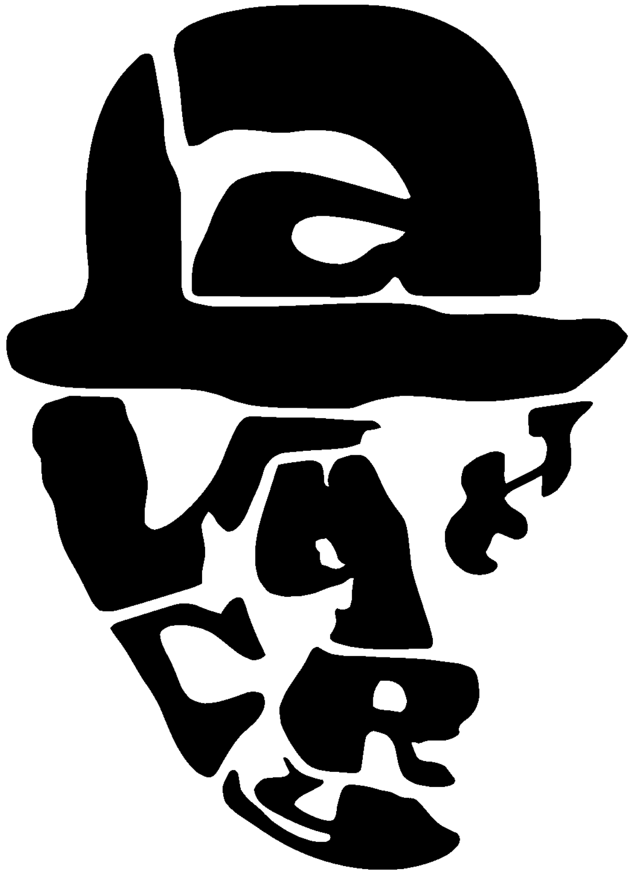


(b)

Figure 16: A “graceful” dancer in (a), and an airplane flying “like a bird” in (b).



(a)



(b)

Figure 17: A calligraphic packing of “GI 2007” into the Graphics Interface logo in (a), and Charlie Chaplin depicted using the words “laugh” and “cry” in (b).