Digital Micrography

Ron Maharik
Mikhail Bessmeltsev
University of British Columbia

Alla Sheffer
University of British Columbia
INRIA Rhône-Alpes

Ariel Shamir
The Interdisciplinary Center

Nathan Carr
Adobe Systems Incorporated

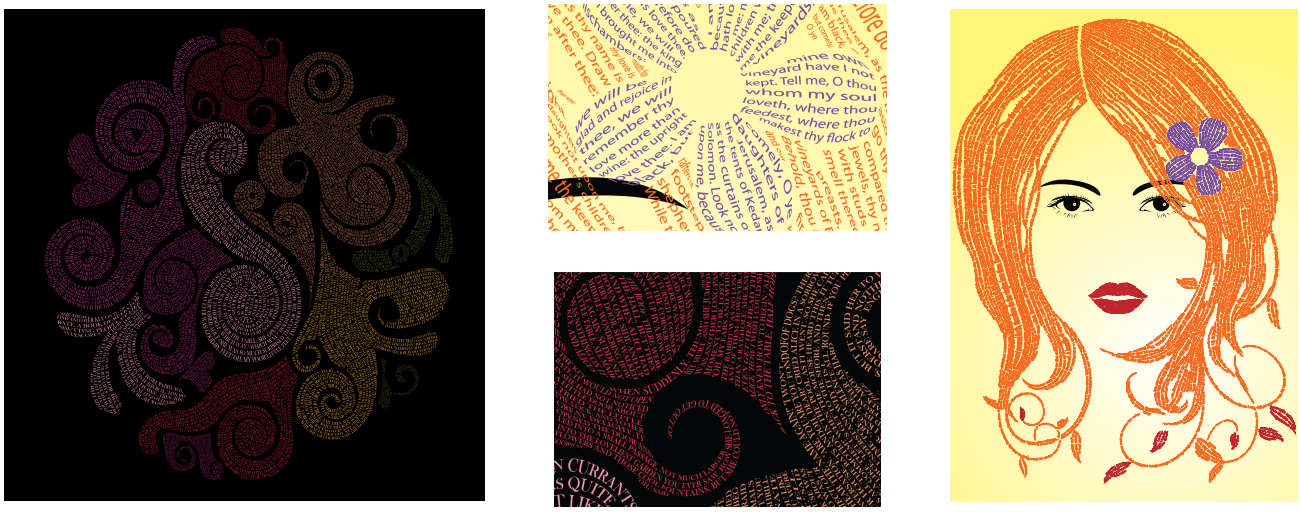


Figure 1: *Micrography images created using our system. Closeups of parts of the images shown in the middle. Left: excerpt from Alice in Wonderland, target size 110x110cm. Right: Song of Songs, target size 42x60cm.* **Please zoom into the images using the digital version to read the fine text. See supplementary material for large images.**

Abstract

We present an algorithm for creating digital micrography images, or *micrograms*, a special type of calligrams created from minuscule text. These attractive text-art works successfully combine beautiful images with readable meaningful text. Traditional micrograms are created by highly skilled artists and involve a huge amount of tedious manual work. We aim to simplify this process by providing a computerized digital micrography design tool. The main challenge in creating digital micrograms is designing textual layouts that simultaneously convey the input image, are readable and appealing. To generate such layout we use the streamlines of singularity free, low curvature, smooth vector fields, especially designed for our needs. The vector fields are computed using a new approach which controls field properties via *a priori* boundary condition design that balances the different requirements we aim to satisfy. The optimal boundary conditions are computed using a graph-cut approach balancing local and global design considerations. The generated layouts are further processed to obtain the final micrograms. Our method automatically generates engaging, readable micrograms starting from a vector image and an input text while providing a variety of optional high-level controls to the user.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; J.5 [Computer Applications]: Arts and Humanities

Keywords: calligraphy, micrography, digital typography

Links: DL PDF VIDEO

1 Introduction

A calligram is an arrangement of words or letters designed to create a visual image. Calligrams enjoy a rich tradition and wide variety of styles limited only by the artist's imagination. As stated by British book designer Thomas James Cobden-Sanderson: "The whole duty of Typography, as of Calligraphy, is to communicate to the imagination, without loss by the way, the thought or image intended to be communicated by the Author". A special type of calligraphy known as micrography (or microcalligraphy) utilizes minute letters to provide a unique interplay between textual content and image - presenting a story or poem at the small scale and forming an image when viewed as a whole. The gap in scale between lettering and image is a defining characteristic of micrography and distinguishes it from other types of calligrams.

Micrography places a large emphasis on the readability of the text, with traditional micrography images, or *micrograms*, typically drawn by professional scribes. While there are wonderful examples of this art throughout history (see [Apollinaire and Greet 1980;

Figure 2: *Examples of artist generated micrograms, by (left to right) G. Apollinaire, G. Klopstock (gilklop.co.il/apage/35286.php), and J. Sibal (jonsibal.com). Used with permission.*

Figure 3: *Choosing a poor text layout such as simply using horizontal lines (left), can create lines that are too short in thin features and do not depict the overall shape. Our layout algorithm (right) better conveys the shape and avoids the creation of short, fragmented, text lines.*

Avrin 1981; Vahe 2009] and Figure 2), they tend to be very rare due to the immense time investment required to create them, as well as the technical precision necessary to scribe legible text at the desired minuscule resolution. In this paper we present a method for creating digital micrograms starting from a given text and an appropriate input vector-graphics image. Our goal is to automate the highly technical and time consuming components of microgram creation while providing optional high level tools for user control. The micrograms created using our system effectively convey the input images using readable text and are visually appealing (Figure 1).

Not every image can be used to create a meaningful microgram. Micrograms usually contain several continuous regions of text conveying large image components, as well as individual text lines representing image strokes. Some also contain non-textual elements (Figure 2, center) enhancing the visual appeal of the image. Hence, the natural input to our system is vector graphic images where such regions and lines are well defined. More complex images can be simplified and converted to such a representation using standard tools [Adobe 2010]. In the remainder of this paper we work exclusively with vector representations, which are practically resolution independent. Reproducing such vector art on various media and for various sizes and colors is a separate topic with its own literature [Sharma 2002; Ostromoukhov and Hersch 1999; Knuth 1997].

Traditional micrograms were rendered in black and white with text alignment used as the main tool for conveying the structure of the image [Avrin 1981]. Today, we can use color, but textual layout is still the main challenge in creating micrograms. The dual nature of a microgram dictates two sets of requirements: one concerning textual readability and the other concerning image recognition and aesthetics. Hence, our goal is to define a method for text layout that depicts the input image using a set of readable text lines, in an automatic or user-guided setting.

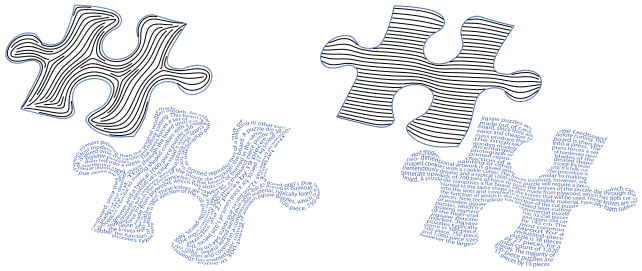


Figure 4: *The streamlines of a smooth vector field aligned with region boundaries (left) capture the shape of the region, but can have high curvature and lack coherence, making the text hard to read and the image less appealing. Our layout (right) selectively relaxes the alignment requirement to provide both a readable text and a visually appealing representation of the input image. Text from "Jigsaw Puzzle" (Wikipedia)*

Readability of text is translated to text flow; the reader must be able to follow the lines of text in a smooth and natural manner [Zachrisson 1965; Tondreau 2009]. At the single line level this means that lines should have *low curvature* with no sharp changes in direction. They should also have some minimal *length* to support readability (Figure 3). At the region level this means some minimal *spacing* should be preserved between lines of text. At the whole image level this means that lines inside each region and text across regions should follow a natural *ordering* for readability, (e.g. a general left-to-right, top-to-bottom ordering in English). Note that ordering requires a *coherent* line orientation with smoothly changing line directions within each region, such that a natural ordering is indeed possible (Figure 4).

To depict a given image the text should adhere to the shape and shading of the image. This is similar to the use of strokes or lines in sketches. Since the text layout defines the outline of the shape, the text line directions should *align* with the image region boundaries when possible (Figure 3). Specifically, acute angles between boundaries and text directions should be avoided since they create an aliasing effect. *Low-curvature, coherent*, and well *spaced* lines are important not only for readability but also for visual appeal. When color or tone is not used, the text direction along shared boundaries between adjacent image regions should be different or even orthogonal to make the boundaries clear and each of the regions *distinct* (Figure 10 center). Our challenge is to design text flow that conforms with this set of sometimes contradictory requirements, some of which are more critical than others, some local and some global.

The streamlines of a smooth vector field aligned with region boundaries provide aligned, smooth and well-spaced lines for text layout, thus serving as a natural starting point for our method. However, forcing alignment everywhere along the boundary will by necessity create areas of high curvature, and depending on the choice of vector field formulation used can introduce singularities leading to loss of coherence (see Figure 4). Our challenge is therefore to balance alignment with readability by selectively relaxing the alignment constraint. One possible methodology to achieve this is through outlier-robust optimization techniques, which allow one to balance a set of constraints, selectively relaxing hard-to-satisfy ones. However, these methods are typically numerically expensive and would require solving a large non-linear system throughout the image regions. Instead, we observe that in our settings we can explicitly compute the set of satisfiable alignment constraints before solving for the field inside the region. In our case a satisfiable set of constraints is one providing the desired balance between alignment and readability, e.g. for the puzzle piece in Figure 4 our method only en-

forces alignment along the more horizontal sides of the piece while for the part in Figure 3 alignment is relaxed at the two tips.

To define these constraints we sample the image region boundaries and build a graph whose nodes are the sample points and whose edges contain weights defining different relations between these points. This construction reduces our boundary condition setting problem to a graph-cut, albeit one with a mixture of positive and negative weights. We use a stochastic quadratic programming search algorithm to find a solution, then compute a smooth vector field within each region which satisfies the selected alignment constraints. The streamlines of the computed fields (Figure 4 right) satisfy our set of requirements providing the basis for the layouts. Next, we trace these streamlines inside each region using a modified tracing method adapted to our needs. We then compute a consistent ordering between them within and between regions which optimizes the overall text readability. In the general case an optimal ordering defines the shortest Hamiltonian path, with respect to some distance metric, and is NP-Hard to compute. For text layout purposes we leverage the left-right, top-bottom layout conventions to obtain an acceptable ordering in a reasonable amount of time. Lastly, we place text along the ordered set of lines, adaptively scaling and warping it as necessary to create well-defined and evenly shaded regions.

Our method can provide a fully automatic solution for a given input text and vector image. In addition, we provide a user interface to allow artists more specific control over the layout. This includes specifying layout direction preferences, font size and type, and other specific controls on the different requirements for various artistic effects.

2 Previous Work

Text Art: The labor intensive nature of artistic expression using text art is well recognized. For this reason text form generation, or text art, has garnered attention in computer graphics in a number of different areas. For example, technologies for the placement of text along user-specified paths have been well explored [Surazhsky and Elber 2000; Asente 2010] and are widely available in commercial vector design packages such as Adobe Illustrator [2010]. While we rely on these tools in our work, automating text placement alone is not sufficient for creating sophisticated micrography images. Specifying dozens and sometimes hundreds of lines to convey a complex image and/or a long text is a daunting task to perform manually. More significantly, designing a suitable, ordered set of lines for simultaneously conveying both a readable text and a complex image is far from trivial (see Figures 3 and 4).

Computer aided design of text-based art-forms has been explored in a number of different contexts. The work of Xu et al. [2010] showed how to generate *structure-based* ASCII art by careful analysis of line structures. Such work aims at approximating shape with a limited set of symbols under rigid rules for alignment and position but does not attempt to place meaningful text in a legible form. Macroscopic calligram generation, as explored by Xu and Kaplan [2007a], packs and warps a single word or a small set of text characters into a given shape. In contrast our method specifically focuses on *microcalligraphy*, where entire text passages are contained within the generated art. A number of software packages produce readable text in the form of grid-aligned text-mosaics [Helmond 2010; Froumentin 2010]. Micrography draws from a much broader artistic palette using shape at both the macro image-level and the micro font-level. Our system aims at delivering this greater degree of freedom.

Non-textual layout: Our goal of laying text inside a set of regions is related to a number of other layout problems, such as design of ornamental patterns, maze creation, NPR rendering, and hatching. The criteria for designing ornamental patterns are quite similar to those for laying out text. Wong et al. [1998] proposed a greedy layout method which has limited applicability and raised the need for a global layout strategy in their future work section. Our work introduces such a global strategy in the context of text layout. Computer generated mazes [Pedersen and Singh 2006; Xu and Kaplan 2007b] pack evenly spaced paths into a given region but contrary to our goal of readability, they encourage such paths to wind and bend. Lastly, hatching and NPR rendering methods use shape information to place strokes that best convey a 3D input shape [Praun et al. 2001; Jodoin et al. 2002]. Our input is a 2D image, and we aim to convey not only individual strokes but also use readable text to shade whole image regions.

Layout using vector fields: Smooth fields and particularly direction fields have played an important role underpinning many layout tasks such as texture synthesis, uv parameterization, mosaics, and quad-remeshing [Xu et al. 2009; Ray et al. 2009; Li et al. 2010; Bommes et al. 2009; Palacios and Zhang 2007]. In all of these settings the boundary conditions for the field are fixed, and design flexibility comes from balancing smoothness, curvature and singularity count. Ray et al. [2008] and Crane et al. [2010] construct direction fields with a user-defined set of singularities. Fisher et al. [2007] and Bommes et al. [2009] attempt to minimize the occurrence of singular points without direct user control. Ray et al. [2009] provide a trade-off between curvature and singularity count. In our setup, singularities are undesirable as they reduce the coherence of the resulting text layout, but may be unavoidable when other, more critical requirements are enforced. Contrary to all those settings we have the additional freedom to modify boundary conditions, trading alignment for lower curvature and singularity count.

The image depiction and sizing requirements raised by our text layout are quite similar to the criteria optimized in the layout of structured textures [Xu et al. 2009] and anisotropic image mosaics [Li et al. 2010]. Both methods use smooth vector fields as the major layout tool. Xu et al. [2009] align an anisotropic texture with feature lines on a 3D surface, while Li et al. [2010] align a vector field with manually selected directions approximating an image gradient. Both methods try to satisfy the given alignment constraints in a least-squares sense, typically at the expense of introducing high curvature and singularities into the resulting fields. Using such fields for text layout creates sub-optimal results (e.g. Figure 14). In our setup we use the additional flexibility of designing the boundary conditions for the fields based on a combination of considerations including alignment, coherence and low curvature. We selectively relax the alignment requirement to allow for the desired low level of curvature and coherence, arriving at vector fields better suited for our task (Figures 12 and 14).

3 Algorithm Overview

Preprocessing: Our micrography algorithm uses as input a vector graphics image and a given text. The input image can contain three components: regions that need to be filled with text, individual strokes used directly for lines of text, and additional graphical elements which are left untouched by our method (see Figure 10 for some examples). In addition to the colors already present in the images, users can specify additional parameters such as font type and size. They can also influence the text layout by specifying local alignment preferences.

If the user does not specify a typeface size, we set it based on the ratio between the area of the regions being filled and the area of a

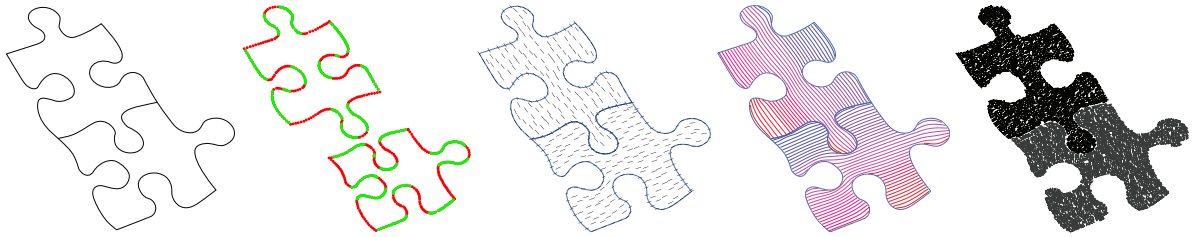


Figure 5: *Overview: Given the input vector image (a), we first compute the optimal boundary conditions for a desired vector field expressed via region boundary alignment constraints (b), where green points are labeled 1 (or aligned) and red are labeled 0 (or non-aligned). We then calculate a smooth vector field (c) that satisfies these boundary conditions. Next, we trace the field streamlines, orient and order them, blue-to-red coloring visualizes line order inside each region (d); and finally place the text along them (e).*

rectangle used to layout the user given text using a default typeface size. This value is clearly just an estimate as lines can never be packed to fill the regions with perfectly even spacing. The final text placement stage adjusts both font size and other parameters to obtain an accurate fit between the text and the generated set of streamlines.

Vector field design: The first and main stage of our method computes a vector field across each region such that the streamlines of the field define the line layout (Section 4.2). To compute the desired field we first design the suitable boundary conditions for it (Section 4.1, Figure 5b). Specifically, we specify for every point along the boundaries of each region whether the vector field should be locally aligned with the boundary or not. These alignment constraints are computed so as to balance boundary alignment with curvature, coherence, and streamline length considerations. Given the computed alignment constraints we solve for a smooth 2-RoSy vector field [Palacios and Zhang 2007] inside each region which satisfies these constraints in a least-squares sense (Section 4.2, Figure 5c).

Line tracing: We trace a set of desired streamlines in the generated field using a modified line tracing mechanism that aims to avoid the formation of short lines, optimize spacing, and prefer coherent end-point placement (Section 5). We then orient the lines and find a traversal, or layout, line order (Section 6, Figure 5d). Finding an optimal traversal order with respect to a distance metric is equivalent to solving the shortest Hamiltonian path problem, which is known to be NP-Hard. Our method therefore relies on the coherence of the traced streamlines to obtain an order that agrees with traditional ordering conventions.

Text Placement: Finally, the text is placed along the lines with user-provided color, font, and size information, and adjusted to ensure a good fit over the entire image as well as at each individual line, in terms of both look and readability (Section 7, Figure 5e).

4 Vector Field Design

The goal of this step is to design a vector field suitable for our needs. Since we aim to use the field streamlines for laying out the text, a correctly designed field should facilitate the formation of streamlines that *a priori* satisfy the image and readability requirements discussed above. While some requirements such as spacing are only relevant for the streamline tracing and subsequent steps, others can be taken into account in the field design step. These include low curvature, minimal lengths, region boundary alignment, coherence, and distinction.

While various techniques exist for computing smooth vector fields

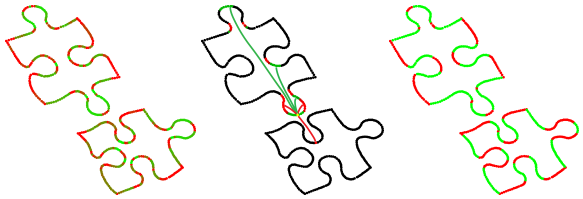


Figure 6: *The graph constructed for the jigsaw example in Figure 5. Left: a depiction of the attraction to the 1 (aligned) label for each vertex in the two regions, visualized by a red to green range, with green representing maximal attraction. Center: a subset of the weighted-edges graph from one vertex node to its neighborhood, vertex and edge color show similarity weights, with negative weights in red, positive in green. The full graph contains such edges from all vertices. Right: one of the local optimum solutions encountered during the stochastic search. The best labeling found is shown in Figure 5b.*

that satisfy conditions such as alignment or distinction, such fields are not necessarily coherent or have low curvature. Boundary aligned smooth fields can still contain high-curvature regions and singularities, which prevent intuitive streamline ordering (see Figure 4 left). Singularities in particular correlate with undesirable artifacts in almost every layout application, and our case is not different. Our method is capable of handling singular points (see Figure 11 lower right), but these should be avoided when possible. Many previous works focus on reducing the number of singularities or providing user control over their location and characteristics [Palacios and Zhang 2007; Crane et al. 2010]. In contrast, we aim to avoid such singularities *a priori* by defining suitable boundary conditions. Specifically, we aim to selectively relax the boundary alignment constraints in parts of the boundary to reduce both singularity counts and curvature, while simultaneously taking into account length and distinction considerations. Such selective relaxation was not, to our knowledge, addressed by previous vector field design research.

We require a method of controlling boundary conditions, such that we enforce alignment where it is critical and relax it elsewhere. One possible approach is to use an iterative constraint reweighing scheme, somewhat similar to Xu et al. [2009], or a more formal outlier-robust optimization approach that balances smoothness and alignment. However, such approaches can become numerically expensive for large regions and can be hard to control. Instead, we observe that in our setup the process can be broken into two steps, where we first compute a suitable set of boundary conditions and then use a simple linear solver to obtain the vector field throughout.

4.1 Boundary Conditions Design

The only type of boundary condition, or constraint, we aim to use for the field design is alignment with region boundaries. To design the conditions we therefore only need to decide for each point on the boundary: should the text be aligned to the boundary at that point or not? To make these decisions we use a discrete setup by sampling the boundaries with vertices, and aim to assign a boolean value per vertex: *0* for non alignment or *1* for alignment.

We formulate this assignment as a global labeling problem for all vertices along region boundaries. There are two factors that affect the assignment of each vertex: the individual labeling preferences or attraction of the vertex based on global constraints, and the similarity or dissimilarity preference between pairs of vertices in the same neighborhood. Such problems are often formulated as a graph-cut by representing each vertex as a node in the graph and the pairwise preferences as weights on the edges between nodes. We build such a graph and add an edge between the nodes of each vertex and all other vertices that are "visible" from it, meaning that the line between them does not intersect the region boundaries (see Figure 6 center).

Weights: The edge weights reflect the degree of similarity or dissimilarity we aim to have between the node labels. Since we want a coherent, low-curvature field we would expect the desired field at nearby vertices to have similar directions. Consequently, if nearby vertices have similar tangents, one would prefer them to have the same label, while if the tangents are orthogonal one would prefer them to have different labels. This requirement can be encoded as a positive or negative weight assignment w_{ij} on edges between the nodes i and j:

$$w_{ij} = F_a(a_{ij}) \cdot F_d(d_{ij}) \tag{1}$$

where d_{ij} is the distance between the vertices of nodes i and j, and a_{ij} is the angle between their normals. The angle factor is a hyperbolic tangent function that smoothly varies from -1 to 1 depending on the angle:

$$F_a(a_{ij}) = \tanh(\frac{4}{\pi}(|\frac{\pi}{2} - a_{ij}| - \frac{\pi}{4})) \tag{2}$$

With appropriate scaling, this results in a smooth transition from a weight of 1 for perfectly aligned tangents to -1 for orthogonal ones. To make the correlation between vertex labels become weaker with distance, the distance factor is set to a Gaussian function of $\tilde{d}_{ij} = \max(d_{ij} - d_{min}, 0)$. Both these and other distances in the formulation below are normalized by, or expressed as a multiple of, the font size in each region. We use $d_{min} = 5$ as the minimal acceptable line length, where we want the labeling correlation to be strongest. We therefore have

$$F_d(d_{ij}) = e^{-\tilde{d}_{ij}^2/\sigma_d^2}$$

with $\sigma_d = 10$ providing a good balance between coherence and individual vertex alignment preferences discussed below.

If we have more than one region and use the direction of text as a method for distinction, as we do in the jigsaw (Figure 5), we duplicate the nodes of boundary vertices on each shared boundary and use one node per region. We then introduce an edge with a maximal negative weight of $w_{ij} = -1$ between them. Otherwise the graph-cut problem is solved independently per-region.

Attraction: As a default, a labeling of 1 is preferred for each node, with the exception of nodes corresponding to vertices at sharp corners where no meaningful tangent direction is available. However, the strength, or weight, of this attraction to 1 can vary based on global considerations and specifically the impact of alignment choice on the length of traced streamlines, and consequently text lines. While it is hard to predict the length of a streamline aligned with a boundary at any given point a good estimate of the length of a non-aligned, or orthogonal streamline is the feature diameter at that point. Hence, since we aim to reduce the number of short streamlines, the attraction should be inversely proportional to this diameter.

One way to measure the diameter and define the attraction strength is to use the local feature size [Amenta and Bern 1999] and the degree to which the tangent at the vertex is aligned with the local feature skeleton. However, robustly computing local feature size, the skeleton and the matching between the boundary and the skeleton are non-trivial tasks. Instead, similar to Shapira et al. [2008], we use an approximate "diameter" measure by evaluating the distance and degree of alignment between the normal at each vertex and the normal at the "opposite" boundary of the region. For node i representing vertex v_i, we define the opposite boundary vertex as the closest intersection of the boundary with a ray emanating from v_i in the opposite normal direction. The distance to the intersection approximates twice the local feature size, while the angle between the normals (or tangents) at v_i and its opposite vertex reflects the degree of alignment with the local skeleton. To avoid inaccuracies due to slight variations in a normal, we consider a cone of rays around the opposite normal of v_i and not just a single ray. We trim the resulting set of opposite vertices to remove outliers in terms of distance and normals, and use an averaged distance d_i and average normal direction a_i in the attraction weight α_i for node i (Figure 6 left):

$$\alpha_i = F_a^+(a_i) \cdot F_d^+(d_i) \tag{3}$$

This formulation is very similar to the inter-node weight function in Equation 1 with two changes. First, we have $F_a^+ = max(0, F(a_i))$ as we aim for default non-negative attraction. Second, to encourage alignment even for large features we use a larger value of $\sigma = 20$ for the distance Gaussian distribution $F_d^+(d_i)$.

To make the comparison between the quadratic similarity-weight component and the linear attraction one independent of sampling density we scale α_i by $\|W\|/\|\alpha\|$ where W is the symmetric weight matrix and α the attraction vector.

We finally arrive at a classical labeling problem based on graph cut where we search for a minimum of the quadratic functional:

$$\min \sum_{ij} w_{ij}(l_i - l_j)^2 + \omega \sum_i \alpha_i(1 - l_i) \tag{4}$$
$$\text{subj. to } 0 \le l_i \le 1$$

The coefficient ω controls the tradeoff between coherence and attraction and has default setting of $\omega = 1$.

Solving: If all w_{ij} were positive, Equation 4 would be a standard convex problem. However, with negative weights the problem becomes much harder. In fact, if the attraction component is zero, by flipping the negative and positive weights the graph-cut problem becomes one of finding a maximal cut, one of the twenty-one classical NP-complete problems [Karp 1972]. In our setup due to the weight settings the problem is somewhat more constrained, since our solutions tend to have similar labels for consecutive boundary vertices with a small number of discontinuities where the label changes (see solutions in Figures 6 and 5). Consequently, we are able to use a tailored stochastic search approach to compute a good solution.

Specifically, we generate a set of randomized initial guesses and search locally for the best solutions near these guesses. For the local optimization we use an active set based quadratic programming approach [Gill et al. 1981], which is well suited for this type of problem. For the first initial guess we use a neutral setting of $l_i = 0.5$

for all i. Intuitively, given this setting the method is most likely to move the labels according to the similarity weights, keeping them together when they are similar and separating them when not. Consequently, we use a local randomization strategy, where each new initial guess is alternatively generated either from the best global solution so far, or from the most recent solution. To generate another guess from a given solution, we randomly select consecutive portions along each region boundary and invert their labels. This strategy is suitable for our problem as in the final labeling there are very few discontinuities. The location of the inverted region and its length are selected randomly and with equal probability. Ten to twenty iterations are typically sufficient to arrive at a good result.

Since the labeling is obtained using a numerical approach we can have non-integer labels. Thanks to the similarity weighting the number of such labels is typically very small. To remove those from consideration we clamp all labels smaller than $1 - \epsilon$ to zero.

4.2 Computing the vector field

Once the boundary conditions are computed, we solve for a smooth field within each region. Several types of fields have been used successfully for layout tasks, including direction and orientation fields, N-symmetry fields and general vector fields. We chose to follow the method of Palacios and Zhang [2007] by solving for 2-rotational symmetry (2-RoSy) fields, which exhibit a high degree of smoothness and are very efficient to compute. A possible alternative is to use advanced field design schemes such as those of Ray et al. [2009] or Fisher et al. [2007]. Methods that avoid singularities at all costs [Crane et al. 2010] are less suitable for our needs as they tend to compensate for those with an undesirable increase in curvature.

We triangulate each region and compute the field indirectly by solving for its corresponding *representation vector field* R:

$$R_i = \begin{pmatrix} cos2\theta_i \\ sin2\theta_i \end{pmatrix}$$

where $\theta_i \in [0, \pi)$ is the angle of the 2-RoSy field at vertex i in the triangulation. The representations R'_l of the tangents at the region boundary vertices labeled as 1 by the boundary constraint design algorithm are added into the system as soft constraints on the corresponding representative field values R_l with unit weights,

$$\min \sum_{ij} \|R_i - R_j\|^2 + \sum_l (R_l - R'_l)^2$$

Users can control the field design either by directly specifying directional constraints inside a region or by modifying the boundary conditions. We found the latter to be more intuitive. Boundary conditions can be edited by adjusting the weight ω that controls the balance between attraction and coherence, or by specifying a subset of labels l_i as hard constraints (see Section 8 for some examples) and then solving for the rest of the labels.

We found that the fields generated using our two-step approach are singularity free and low-curvature whenever a solution exists that does not violate other, more critical, requirements such as user constraints or line length. An example of a singularity generated due to user-imposed alignment constraints is shown in the lower right corner of Figure 11, where users required the text to be aligned with both the top and the sides of the head. Figure 1 (left) contains a few regions where coherence and low curvature cannot be satisfied without introducing very short lines. Consequently, higher curvature or one or two singularities are introduced automatically to allow for longer streamlines.

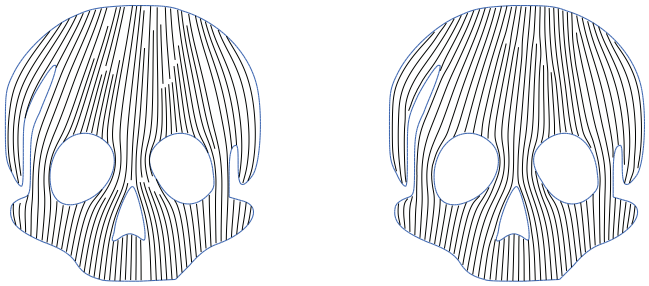


Figure 7: *Basic tracing mechanism creates both short and unorganized lines (left). Using our strategy the lines are better organized for text flow (right).*

5 Tracing Lines

Since the generated fields satisfy the coherence and flow requirements *a priori*, we can use a fairly simple tracing approach [Jobard and Lefer 1997] to generate the desired set of lines with a few changes directed at improving readability. To maximize text readability we aim to mimic, as much as possible, the reading flow of rectangular text regions. This can be translated to two main objectives: first, that lines should begin and end close to each other in a direction orthogonal to the field (i.e. as if they are "flushed" together to a margin), and second, that line length should be as long as possible. Consequently, we try to trace long streamlines and cluster their end points (Figure 7).

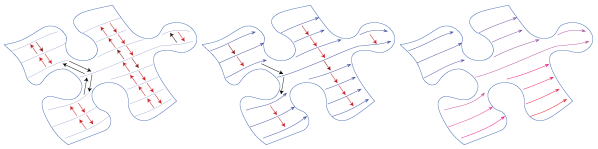
Finding seeds: Similar to previous tracing methods [Jobard and Lefer 1997; Chen et al. 2008] we use a priority queue of potential seed points for new streamlines. At each iteration, the next seed is removed from the queue and if it is sufficiently distant from the boundaries and nearby streamlines, it is used to trace a new streamline. During tracing, points at a fixed distance in a perpendicular direction to the streamline on either side are added to the queue as new seed candidates.

The seed priority function is adapted to our settings. To cluster end points together we prefer to select seeds above or below the end points of existing streamlines in the orthogonal direction of the streamline. Hence, seeds have higher priority the closer they are to an existing end point.

To ensure good approximation of the region boundary we initialize the queue with seeds placed near boundary vertices where the field is aligned with the boundary (at an offset equal to half of a line height). Later, we will grow such seeds in both directions. The left-most such point is selected as the first seed to facilitate a readable order, with a left-to-right convention. To create a continuous flow of lines, the priority of the remaining initial seeds is set to be lower than the priority of the seeds generated during tracing of streamlines.

Tracing streamlines: We trace streamlines from seeds using standard line integration with the midpoint rule. Lines are terminated whenever they reach the boundary, come within a minimal distance to another streamline, or turn sharply, as streamlines with high curvature are unsuitable for placing text. We add the turning test as sometimes, due to user preferences, the field generated does not have sufficiently low curvature.

Since in our setup the streamlines are a tool rather than a goal in themselves, we allow them to deviate slightly from the field direction. This is done by modifying the field during tracing to reduce aliasing and extend the line lengths. We augment the input field with a weak field that repels streamlines away from the boundary



(a) (b) (c)

Figure 8: *Orientation and ordering demonstrated on a subset of streamlines: (a) initial adjacency graph, with edge color representing weights (b) spanning forest of oriented lines, with connection edges highlighted in black (c) final order of lines (blue to red).*

and existing streamlines, and decays exponentially with distance. This field is weak enough to only affect tracing of streamlines that are near and parallel to the boundary or to previously traced streamlines. This field prevents premature termination of streamlines as they are weakly repelled from boundaries and other streamlines.

Post processing: To finalize the set of streamlines we perform a few cleanup operations: stitching together nearby tangentially aligned streamlines, aligning nearby start and end points in the orthogonal direction, and discarding very short lines. A comparison between basic tracing [Jobard and Lefer 1997] and our result is shown in Figure 7.

Smoothing and Enrichment: Depending on the setup, we may want the spacing between the streamlines to be uniform or not. Uniform spacing results in uniform region shading as seen in the bear and skull examples in Figure 11, while non-uniform spacing can add richness to the output (e.g. girl image in Figure 10). To create the final spacing we use a weighted smoothing approach, iteratively moving streamline vertices toward the weighted average of their closest points on the two neighboring streamlines in the orthogonal direction. For non-uniform spacing, we use randomized weights when smoothing.

6 Line Orientation and Ordering

Laying out text along a set of arbitrarily oriented streamlines requires selecting a text orientation along each line and computing a line ordering.

Orientation: A basic requirement for intuitive line orientation within each region is that adjacent parallel lines have the same orientation. Depending on the curvature of the lines, this requirement may not always be satisfiable. For instance, consider one line that bends around another as on the skull forehead in Figure 11. Our method aims to find a consistent global orientation that is similar on adjacent lines, when possible.

We define a weighted graph where each node corresponds to a streamline and directed edges connect each pair of adjacent parallel streamlines. The weight associated with each edge from one streamline to another is set to be the percentage of the length of the first streamline where the two lines are adjacent. Note that two adjacent streamlines will have two directed edges, but these edges may have different weights (Figure 8a).

The orientation within each region is defined using a prioritized breadth-first traversal on this graph. To initialize the traversal we select a line that has a clear orientation preference, i.e. it is largely horizontal, and has sufficient length (at or above the average line length in the region). The initial line is oriented left-to-right and this orientation is propagated through the graph using a breadth traversal order based on the edge weights. If no line has a strong left-right preference, e.g. if all streamlines are vertical, we use a bottom-up orientation instead, consistent with a left-right line order.

Ordering: The requirement for intuitive ordering involves both region ordering and line ordering within each region. Inside a region, for any two adjacent parallel lines, the lower one with respect to the determined orientation should, whenever possible, come after the upper one. This requirement defines a partial, rather than full, order inside the region, as a streamline can have more then one adjacent line above or below it. Moreover, this partial order can contain loops when lines have high curvature, potentially wrapping around other lines, as is the case in the previously mentioned skull example. To define ordering we must convert this partial order to a full order. In the general case, solving this problem with respect to some metric is equivalent to finding a shortest Hamiltonian path, and is NP-Hard. The algorithm we use is targeted at finding an intuitive, readable order when the partial order is consistent, and a plausible one when loops are present.

To find the ordering we use a subset of the streamline graph defined for orientation, discarding edges from lower streamlines to the streamlines above them. Next, we compute an optimum branching [Edmonds 1967], which defines a spanning forest of the graph (Figure 8b). To create a full order we need to connect the forest into a single graph. We traverse the nodes of all trees to locate the shortest edge connecting two trees and add it to the graph. This process is repeated until all the trees are joined together or no edges linking trees are found. In the latter case we treat each sub-graph as a separate region for text layout.

To traverse and order the combined graph, we pick as a root the left-most orphan node (i.e. one with no incoming edges) with respect to the frame defined by the line orientation. We traverse the graph starting from the root. Each time we encounter a node with more than one parent, we traverse all branches above this node in a left-to-right order and only then traverse the node itself. At each branching point with multiple children we use the same left-to-right order to traverse those (Figure 8c).

To order the regions themselves, we use a similar process, but on a region graph instead of a lines graph. The order here is more vague as we have no clear notion of adjacency or parallelism. Instead region nodes are connected by edges based on global adjacency computed using a Voronoi diagram, and edge weights are set to be inversely proportional to the distance between the regions, and directly proportional to the dot product of the transition direction between regions and the preference direction. The preference can follow a lexicographic left-right, top-bottom order, or some mix of the two directions set by specifying an arbitrary direction vector such as $(1, 1)$. We then use a similar graph traversal mechanism where the local frame is set based on the average of the line orientations in adjacent regions.

To order individual strokes (text lines not included in a region) we can treat each as a stand-alone region, or we can treat them all as a single region. While we cannot claim that the ordering computed is necessarily globally optimal, we found that our results agree, in general, with manual ordering both inside and in between regions.

7 Text Generation

Given the ordered streamlines, we place the text along them using the text engine in Adobe Illustrator. It partitions the text into segments of appropriate size, and provides the tools necessary to map these segments to the curved streamlines as well as myriad options for font styling and colorization.

Any given set of extracted streamlines will exhibit some variation

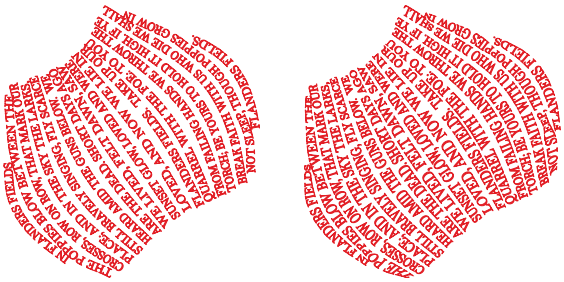


Figure 9: *The text layout for a single petal of the poppy in Figure 10 without text warping (left), and with warping (right).*

in spacing between neighboring curves. To avoid visual gaps in the image, we adaptively warp the height of the text to fill the space more uniformly [Asente 2010]. A similar mechanism is used to adjust text width. The width adjustment is used to achieve two important goals. First, even though an *a priori* estimate is used to set font size in a way that aims to fit the given text exactly into the image, the length of the generated streamlines is never exactly equal to the estimate. Hence, text width needs to be globally adjusted to accurately fit the content into the image. Second, as we want to avoid breaking words in line breaks, we need to adjust the text width to fit each line of text into the corresponding streamline.

Whenever the text lines are not aligned with the boundary of the region, some aliasing artifacts may occur. These artifacts are reduced by our careful vector field design and streamline tracing methods. However, they can still occur as we balance other considerations as well. In addition, along highly curved boundaries even orthogonal layouts can cause some aliasing. To reduce this effect we warp the text along the line using a gradual warp propagation. The warping adjusts the text at the ends of the lines, and propagates the distortion inward to the middle of the line for a smooth effect (Figure 9). While the warping steps improve image aesthetics, resolving aliasing artifacts and promoting uniform shading, they can also reduce readability, especially when the font size is fairly large compared to the local feature size of the image (e.g. in the pupils of Figure 13).

8 Results and Discussion

Throughout the paper we show a diverse selection of micrograms created with our system from graphics and text available freely online. Part of the artistry of micrography lies in selecting source graphics that have the proper amount of detail, along with the right amount of text. If the image is very detailed or the font too large then individual letters begin to play a crucial role; we thus leave the realm of micrography for that of general calligraphy, which is outside the scope of this work. In our system, the choice of sources remains entirely up to the user. Unless stated otherwise, the layouts for all results were computed automatically based on the input graphics and text, which allowed us to rapidly experiment with different fonts and texts to achieve a result that we liked. Please zoom into the images using the digital version to read the fine text. See attached supplementary material for larger images.

The *motif* image in Figure 1 was created using text from Lewis Carroll's "Alice in Wonderland", with user specified font size. Manually tracing the hundreds of lines would be a tedious task, and even just choosing the preferred layout direction per region is difficult. This example also shows the challenge of weighing between readability and image requirements. For most regions a singularity free solution exists, which naturally aligns with the narrow features. The pink region at top-right has six narrow regions with different directions, which are very hard to reconcile. Consequently our solution places one singular point inside the region to obtain what is deemed

Figure 12: *Vector field generated by Xu et al. [2009] (top left) is not suitable for text alignment due to high curvature and singularities. Our method creates a coherent field and readable text layout (top right, bottom left), albeit not symmetric. Adding symmetric constraints using edge weights creates a symmetric field and result (bottom right). Text: Lorem ipsum, target size 20x20cm.*

Figure 13: *A black and white micrograph where region distinction is achieved through variation in font size and type. "Dark Eyes" (translation by Katya from russmus.net), target size 104x54cm.*

an acceptable tradeoff between coherence and alignment. The *girl* image is combined with the words of the Song of Songs. The hair consists of two highly intricate regions with smooth random variation in size used to create the visual effect of hair strands.

Figure 10 shows several other micrograms generated by the system. The *poppy* example uses the famous poem "In Flanders Fields" as a text replicated in every region of the image. The line layout in each region was automatically adjusted to fit the text into the regions, and randomized line thickness is used to enrich the visual effect. The *Mona Lisa* image uses text from "Portrait of a Lady" and exemplifies a black and white classic-style micrography where different regions are visually distinct because of text direction. The *fedora* micrograph shows an example of using text for both regions and outlines creating an interesting visual interplay between the two. The *eyes* example (Figure 13) combines graphical content with a translation of the well-known Russian poem "Dark Eyes" generating a compelling black and white image where font properties are used to visually separate the image regions.

User control: The bear example in Figure 11 shows two results created from the same input using different font sizes, highlighting the flexibility offered to artists when using our tool. In this example our method converts a very simple input into a visually appealing, intricate design by combining a textual micrograph output with a

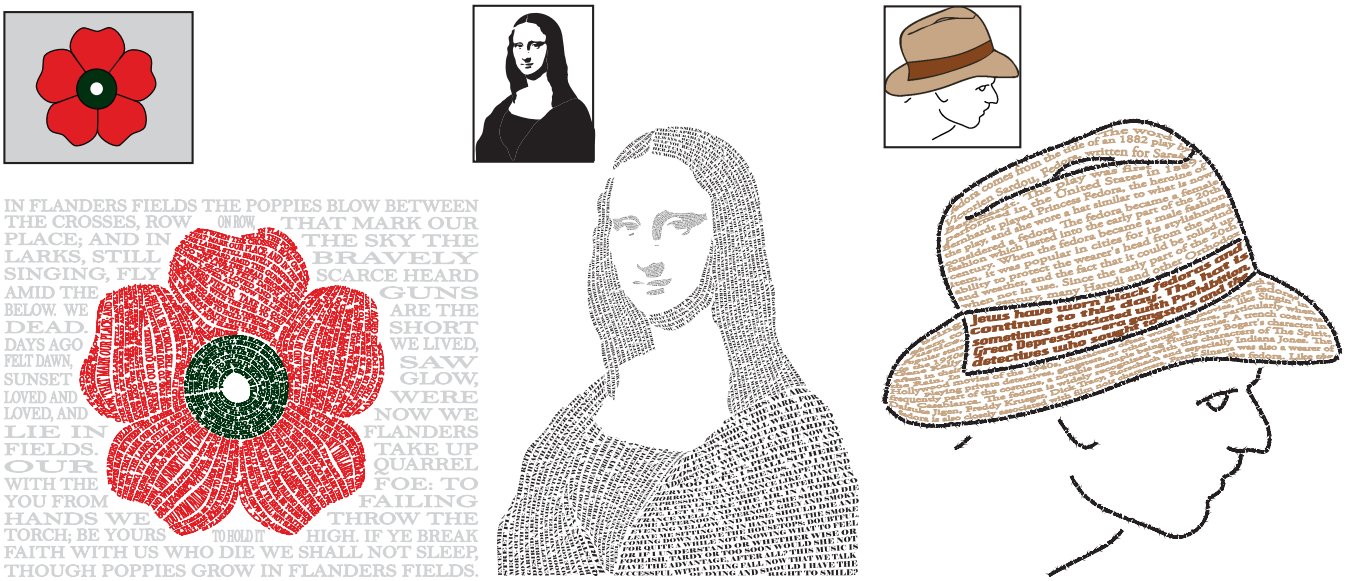


Figure 10: *A variety of micrograms created using our system. Color variation, fonts, and size variability were specified by the user. The original image input is shown at the top. Left: "In Flanders Fields", target size 22x17cm. Center: "Portrait of a Lady", target size 36x45cm. Right: "Fedora"(Wikipedia), target size 26x26cm.*

color layer specified by the user. Such effects are very easy to apply as our output results are naturally created as vector graphics, which can be easily edited. The *insect* examples show variations in both color and typeface.

We considered a number of mechanisms for users to control the design of boundary conditions and consequently the text vector field, such as a direct editing interface for the different parameters, an a posteriori vector field editor [Palacios and Zhang 2007], or selection among different local minima of the QP boundary condition setting problem. We found a direct visual interface where users specify preferred (or undesirable) alignments to be most intuitive. Specifically, users can either *a priori* define preferred text directions inside regions or specify them once given an automatically generated micrography image. The specification can be done using two types of notation: sketching preferred directions as strokes inside the relevant regions, or highlighting existing boundaries as ones that should (or should not) be aligned with the field. The user preferences are then added as strong attraction weights for boundary design. The process can be iteratively repeated until the user is satisfied with the output. An example of such user preference is shown on the Jolly Roger flag in Figure 11, where the automatically generated field is better in terms of the criteria we use (top), while the user guided solution (bottom) prefers strong alignment with the forehead.

Comparisons: We compare our field design to two previous works that produce smooth fields. Figure 12 compares our results to XU et al. [2009]. While their field in general is symmetric, it is not suitable for text layout. Our default output results in a readable layout but does not maintain symmetry. After imposing symmetry constraints through similarity weights in boundary condition computation we produce a text layout that is symmetric, but has more curved lines. Figure 14 shows the consequences of using a vector field designed by a recent mosaic layout method [Li et al. 2010] for micrograph layout. As demonstrated the resulting text layout is incoherent, with multiple singularities. Using our approach on a segmented version of the input image creates more coherent results. The black-and-white example in this figure illustrates the use of the distinction constraint to separate the background from the foreground, creating visibly clear region boundaries.

Run times: The majority of the computation time in our current implementation is spent on two stages: first, the boundary conditions computation which uses a MATLAB implementation of quadratic programming optimization, and second, the actual text layout which uses the Adobe Illustrator text engine [Adobe 2010]. The first takes one to five minutes per region, but can be clearly optimized significantly by switching from MATLAB to a C++ implementation. The second takes on the order of two to five minutes for an average sized input. All the rest of the computations take under a minute altogether.

9 Conclusion

We presented a method for computerized creation of digital micrography images, an appealing and intricate text-art form. Manual creation of micrograms like the ones demonstrated in this paper requires both a high degree of expertise and a huge amount of tedious work from users, while using our system such designs can be created from standard vector art in a matter of minutes. In its default setting, suitable for novice users, the method requires no additional input beyond the vector image and text. For advanced users we provide a range of artistic controls such as font styling and local text directions. The visual appeal of the final micrograph will depend on these choices as well as on proper selection (or creation) of the source material.

The key technical component of our work is the introduction of a novel approach for designing boundary conditions for vector fields, such that smooth fields that satisfy those conditions adhere to additional user requirements. In our specific case we require the field to be singularity free and have low curvature. This approach can be potentially used for other applications that use vector-fields as a design tool, such as creation of anisotropic textures and mosaics.

In our current approach, details of the text to be placed are not used to inform the process. The text engine, while powerful, was not designed specifically for this type of task. Better text placement in-

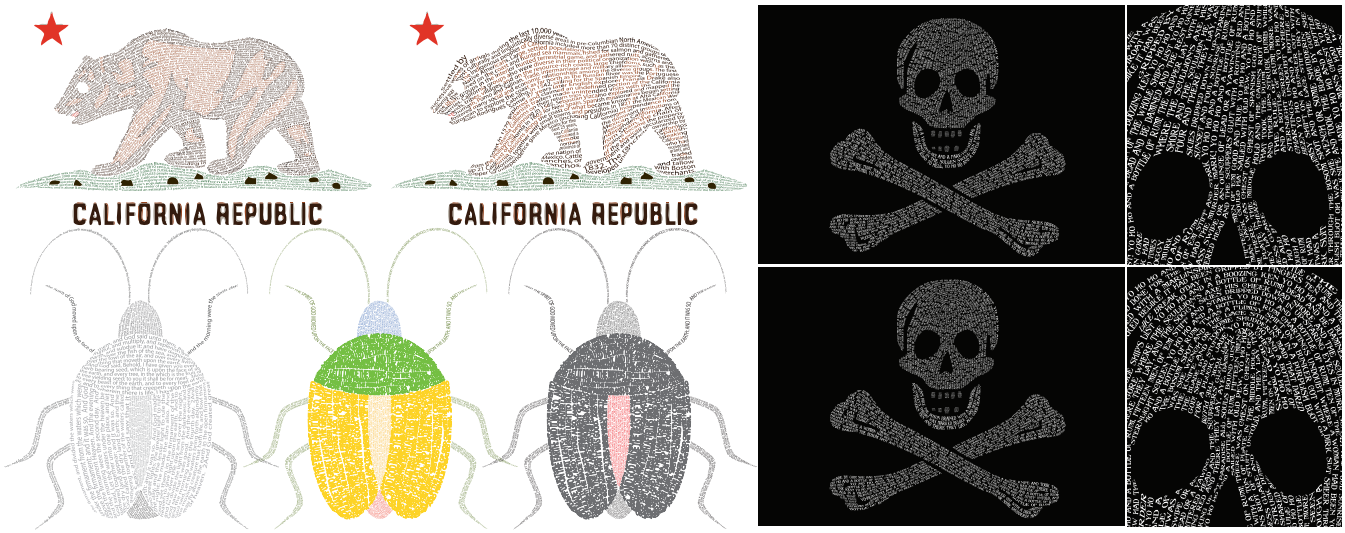


Figure 11: *Change of parameters: the user can change font size, color and typeface, producing endless variations of a microgram. The user can also mark a preferred direction inside a region. The jolly roger flag at the top was created with automatic vector field alignment and has singularity free long text lines. With user preference for horizontal alignment on the forehead the field is less coherent with a wedge singularity formed, but the overall effect is arguably more appealing. Top left: "California" (Wikipedia), target size 28x18cm. Bottom left: Genesis 1, target size 29x32cm. Right: "Dead Man's Chest", target size 223x158cm.*

formed by knowledge of the shape of individual words and letters could be used to reduce artifacts, achieve more uniform density, improve readability, and to insert local calligraphic effects, as authors of traditional micrography often do.

Our work introduces a number of interesting areas for future research. One area would be to map and expand the range of styles supported by the framework: while most micrograms can be generated using piecewise smooth fields, there are examples where this approach requires unintuitive partition of the image into regions. Another is text line ordering: while our method performs quite well on the inputs tested, given the NP-hard nature of the problem addressed it is not guaranteed to provide an optimal result. It would be interesting to consider approximation techniques or other, perhaps perception-based, approaches for ordering a set of text lines such that the reading order is most natural. Accelerating our method is a necessary step for converting it into a more user-friendly tool, and while some steps such as conversion from MATLAB to C++ are straightforward, further speedup of the core algorithmic steps might require more effort. Lastly, it would be interesting to work closely with artists to observe what specific controls of the system they may want in addition to or instead of the ones we provide.

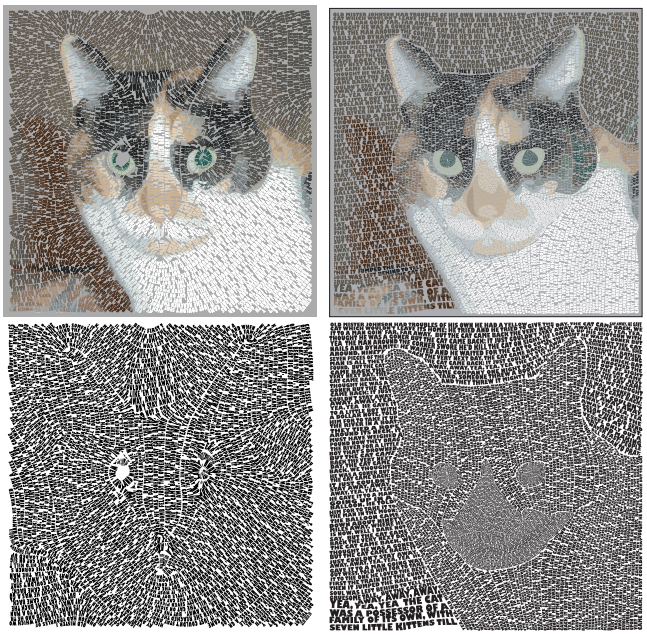


Figure 14: *Using the smooth vector field from Li et al.[2010] to place text creates an incoherent layout (left), as can be seen when colors are removed (second from right). Using our method with a set of extracted regions generates a coherent singularity-free layout, that follows the boundaries whenever possible (second from left). This is even more evident when colors are removed (right). Text: "The Cat Came Back", target size 40x40cm.*

Acknowledgements

We thank Christine Depraz and Ruby Mawira for their help with creating input images, Paul Asente for assistance with Adobe Illustrator, and C. Wang for helping with image conversion. This research was partly supported by Adobe Systems, NSERC, MITACS NCE, and GRAND NCE. We also wish to thank the reviewers for their considerable effort and significant contributions to the finalized version of this work.

The following sources were used in accordance with their respective license: `vector4free.com/vectors/id/277`, `vector.net/2010/summer-girl`, `allfreevectors.com/Free-vector-abstract-illustration!-14584.html`, `russmus.net/song.jsp?song=S:1172202987`, `commons.wikimedia.org/wiki/File:Heraldisch_Lippische_Rose.svg`,

en.wikipedia.org/wiki/File:Fedora_line_drawing.svg, spraypaintstencils.com/a-zlistings/mona-lisa-image.gif, en.wikipedia.org/wiki/File:Flag_of_Edward_England.svg, en.wikipedia.org/wiki/Jigsaw_puzzle, en.wikipedia.org/wiki/California en.wikipedia.org/wiki/Fedora
Sources for Figures 12 and 14 provided by the original authors.

References

ADOBE, 2010. Illustrator CS5 adobe.com/products/illustrator.

AMENTA, N., AND BERN, M. 1999. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry 22*.

APOLLINAIRE, G., AND GREET, A. H. 1980. *Calligrammes : poems of peace and war (1913-1916): A Bilingual Edition.* University of California Press, Berkeley.

ASENTE, P. 2010. Folding avoidance in skeletal strokes. In *Sketch Based Interfaces and Modeling*, Eurographics, ACM.

AVRIN, L. 1981. *Hebrew Micrography - One Thousand Years of Art in Script*. Israel Museum, Jerusalem.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28* (July).

CHEN, G., ESCH, G., WONKA, P., MULLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. *Computer Graphics Forum (SGP) 29*, 5, 1525–1533.

EDMONDS, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standurds 71B*, 233–240.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph. 26* (July).

FROUMENTIN, M., 2010. Textorizer lapin-bleu.net/software/textorizer.

GILL, P. E., MURRAY, W., AND WRIGHT, M. H. 1981. *Practical Optimization*. Academic Press.

HELMOND, A., 2010. Textaizer mosaizer.com/Textaizer.

JOBARD, B., AND LEFER, W. 1997. Creating evenly-spaced streamlines of arbitrary density. In *Eurographics Workshop*, Eurographics, 43–56.

JODOIN, P.-M., EPSTEIN, E., GRANGER-PICHÉ, M., AND OSTROMOUKHOV, V. 2002. Hatching by example: a statistical approach. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, NPAR '02, ACM, 29–36.

KARP, R. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds. Plenum Press, 85–103.

KNUTH, D. E. 1997. *Digital Typography*. Cambridge University Press, New York, NY, USA.

LI, Y., BAO, F., ZHANG, E., KOBAYASHI, Y., AND WONKA, P. 2010. Geometry synthesis on surfaces using field-guided shape grammars. *IEEE Transactions on Visualization and Computer Graphics 99*, RapidPosts.

OSTROMOUKHOV, V., AND HERSCH, R. D. 1999. Multi-color and artistic dithering. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, ACM, 425–432.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. (Proc. Siggraph 2007) 26* (July).

PEDERSEN, H., AND SINGH, K. 2006. Organic labyrinths and mazes. In *Proceedings of NPAR'06*, ACM, 79–86.

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, ACM.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27* (May), 10:1–10:13.

RAY, N., VALLET, B., ALONSO, L., AND LÉVY, B. 2009. Geometry aware direction field processing. *ACM Transactions on Graphics*.

SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. 2008. Consistent mesh partitioning and skeletonization using the shape diameter function. *The Visual Computer 24*, 4 (April), 249–259.

SHARMA, G. 2002. *Digital Color Imaging Handbook*. CRC Press, Inc., Boca Raton, FL, USA.

SURAZHSKY, T., AND ELBER, G. 2000. Arbitrary precise orientation specification for layout of text. In *Proceedings. The Eighth Pacific Conference on Computer Graphics and Applications*, Eurographics, 80 – 86.

TONDREAU, B. 2009. *Layout Essentials: 100 Design Principles for Using Grids*. Rockport Publishers.

VAHE, 2009. Micrography: Text art and typography. gawno.com/2009/05/micrography-text-art-and-typography.

WONG, M. T., ZONGKER, D. E., AND SALESIN, D. H. 1998. Computer-generated floral ornament. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '98, ACM.

XU, J., AND KAPLAN, C. S. 2007. Calligraphic packing. In *Proceedings of Graphics Interface 2007*, ACM, New York, NY, USA, GI '07, ACM, 43–50.

XU, J., AND KAPLAN, C. S. 2007. Image-guided maze construction. In *ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, SIGGRAPH '07, ACM.

XU, K., COHEN-OR, D., JU, T., LIU, L., ZHANG, H., ZHOU, S., AND XIONG, Y. 2009. Feature-aligned shape texturing. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2009) 28*, 5, 108:1–108:7.

XU, X., ZHANG, L., AND WONG, T.-T. 2010. Structure-based ascii art. *ACM Trans. Graph. 29* (July), 52:1–52:10.

ZACHRISSON, B. 1965. *Studies in the Legibility of Printed Text*. Almqvist & Wiksell.