BRIDGES 2025 SUPPLEMENT
Workflow and Algorithms for Holomorphic Mappings for Integrated Garment and Motif Design

Loe Feijs[1,3] and Rong-Hao Liang[1] and Holly Krueger[1,2] and Marina Toeters[4]

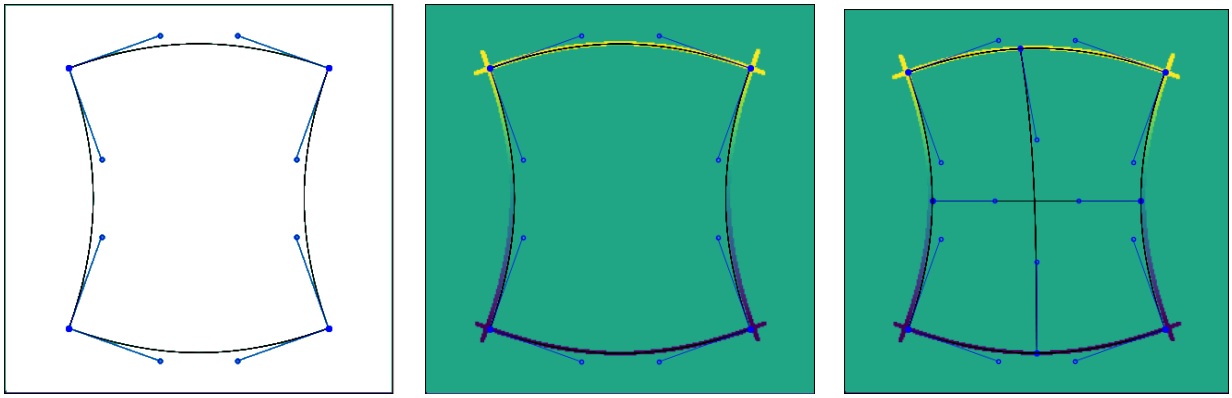
[1]University of Technology Eindhoven, The Netherlands; j.liang@tue.nl
[2]Holly Krueger Design, Amsterdam, The Netherlands; h.l.krueger@tue.nl
[3]Laurentius.Lab, Sittard, The Netherlands; l.m.g.feijs@tue.nl
[4]by-wire.net, Eindhoven, The Netherlands; info@by-wire.net

Abstract

Today's fashion design is based on a separation of concerns such that the geometry of the fabric motif and the geometry of the pattern cut are unrelated. We challenge this approach by developing mathematical tools to morph the motif. The core innovation lies in the mathematical framework, based on holomorphic mappings and harmonic conjugate functions, to map motifs onto arbitrarily shaped panels. This approach, implemented via custom software, allows for seamless motif continuation across complex garment shapes, avoiding cutting through repeating designs. **This supplement** describes the workflow and some algorithmic aspects for designing a single garment panel as described in the main Bridges 2025 paper by the same authors.

Workflow

The starting point of the process is a drawing of the panel, in the simplest case a contour consisting of four Bézier curves, as shown in Figure 1(a). Note the handles of the Bézier curves [5], they can be manipulated interactively, just like in Adobe Illustrator. We use cubic Béziers, so each curve has two handles.



(a) (b) (c)

Figure 1: *Contour definition: (a) Bézier curves, (b) voltages, (c) scaffold.*

The first step after editing is to set the upper and lower contour lines of the panels to specific potential values, which we choose 1 and 0, without loss of generality. The color scheme shows 1 as yellow, 0 as dark purple and an interpolating color for the in-between values, see Figure 1(b). We use the language of electrostatics as a vocabulary [2], stating that we electrified the upper and lower lines with 1 and 0 volt respectively. The side lines of the contour are treated as "insulators" and the image shows this as a potential that changes from 1 to 0 in a continuous manner along each side line. The goal is to find the field lines and equipotential lines so as to form a grid that serves as a basis for mapping the motif later. As a preparation for obtaining meaningful distances, there is an extra step, which we chose to do manually. The two extra

lines in Figure 1(c) form what we call a "scaffold"[1]. In the next step, the scaffold lines will automatically equipped with markers that will serve as the starting points for the steepest descent algorithm that finds the field lines and the equipotential lines.

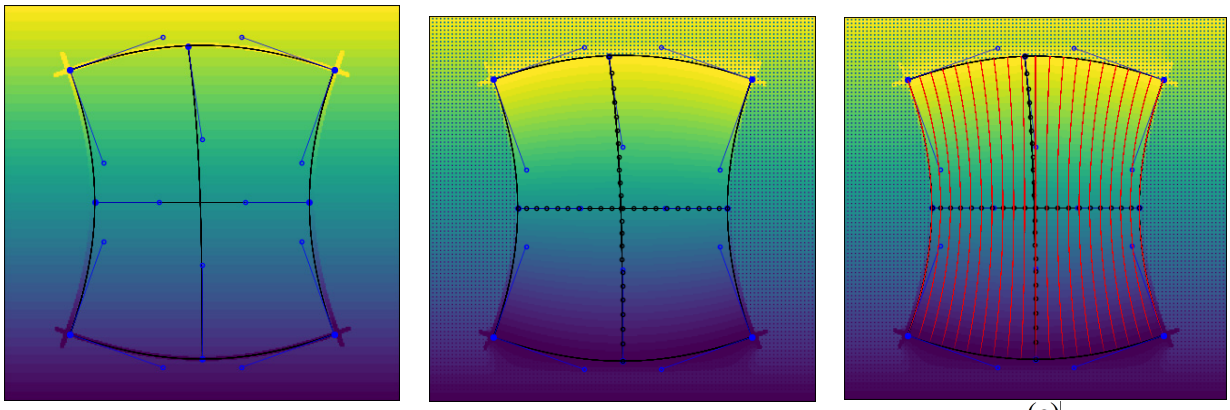


Figure 2: *Computations: (a) initial pot, (b) refined pot, (c) field lines.*

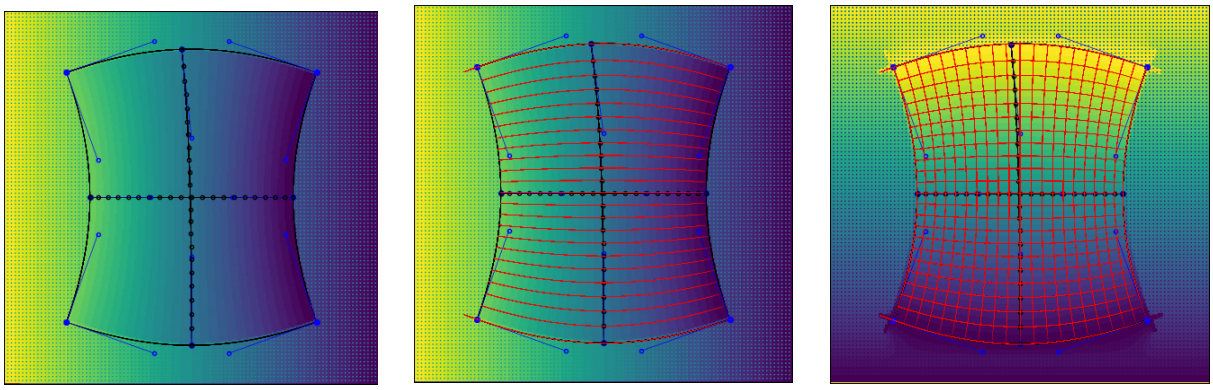
Next we calculate the potential for the entire inner area of the panel. It is already defined by the boundary conditions and we have an algorithm to get the potential everywhere (a relaxation algorithm).

Our algorithm is a bit slow (up to ten minutes on an HP Z-book, depending on the size of the area) and therefore we begin with a good initial guess, shown in Figure 2(a). The potential goes from 1 at the top of the plot window to 0 at the bottom. The window corresponds to an array of size 300×300 (named pot-array), the main data structure for the relaxation algorithm.

In Figure 2(b), the relaxation algorithm is completed and we have the potential for every position. The dotted area is the outside of the contour, where the algorithm produces some values, but these are not very helpful. Also note that the scaffold lines show marker points, which are derived from the potential. The horizontal scaffold line is subdivided into 20 steps, and the vertical scaffold in 22 steps. This means that we will have 21 field lines and 23 equipotential lines. For subdividing the vertical scaffold, the potential defines the marker positions[2], which are separated by steps of 1/22. The field lines are shown red in Figure 2(c) and they go through the markers on the scaffold, as expected.

[1] Instead of using straight lines for the scaffold, which would be sufficient in most cases, I chose to use Bézier curves since at had them already as a class in my code and as editable in my user-interface for the user-defined boundaries.

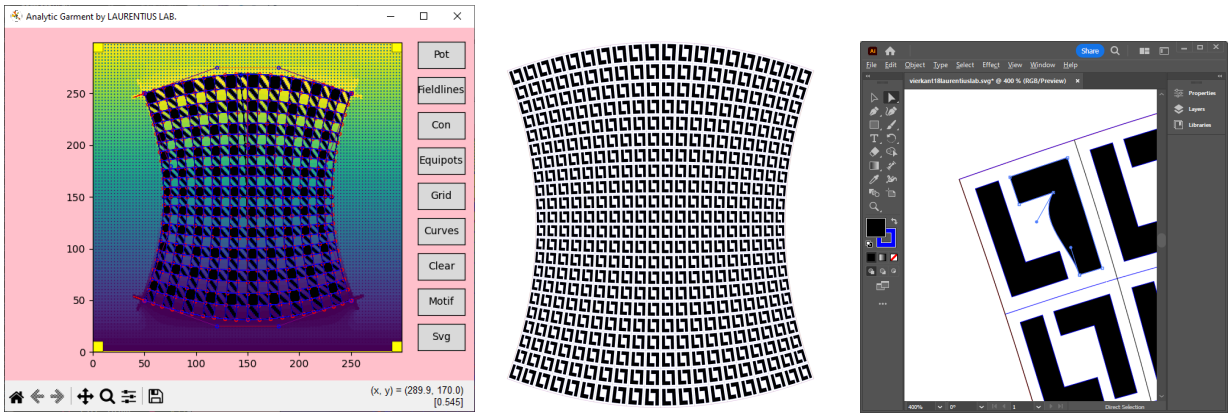
[2] The potential defines the marker positions, by which we mean that once the potential ψ (= voltage) is given at all positions, we can find the points on the scaffold line such that the potential difference is equal for all steps. Since the total voltage difference is 1 volt, we begin at one end, say at 0 volt and then move along the line until we find the point of the desired $1/22$ volt, then we look for the next and so forth. In practice, we move forward in tiny steps and use interpolation within (binary search would be faster, but in view of the much slower relaxation, this once-only electrification of scaffold lines is fast enough). We can find good potential values at *any* point $x + iy$ by interpolating between the values of the (discrete) array-cells.



(a) (b) (c)

Figure 3: *More computations: (a) conjugate harmonic, (b) equipotential lines, (c) both line types.*

In order to do the same for the equipotential lines we use another potential-like array, which we call the *conjugate harmonic of the potential* (con-array). The concept of "conjugate harmonic" is explained in the main Bridges paper and in [1]. If the potential happens to decrease when going vertically, then the conjugate harmonic decreases when going from left to right (for example if $\psi = y$, then potential decreases when going vertically indeed, and then by the Cauchy-Riemann conditions $\partial\chi/\partial x = -\partial\psi/\partial y = -\partial y/\partial y = -1$ and $\partial\chi/dy = \partial\psi/\partial x = \partial y/\partial x = 0$, so $\chi = -x +$ "constant"). We have an algorithm for that too, essentially a kind of numeric path integration with one-pixel steps (no reference available, integration is replaced by summation: I "integrate" by adding $\partial\chi/\partial x$ when making a step in the x direction and $\partial\chi/\partial y$ when making a step in the y direction[3]). The contents of the con-array is shown in Figure 3(a). Using steepest descent [3] in the con-array gives us a bundle of equipotential lines, shown in Figure 3(b). In Figure 3(c) we present both field lines and equipotential lines on a background of the potential (color coded).



(a) (b) (c)

Figure 4: *Final steps: (a) motif morphing, (b) svg file, (c) postprocessing.*

Next the intersection points can be found and thus we have enough for mapping a motif which originates from a rectangular grid onto the new grid. Besides the morphed motif, we also reinterpret the field lines as being built from Béziers [5] (which is in fact an approximation). Each of the 22 segments of each field line is transformed into a tiny (approximated) Bézier curve with two anchors and two control points.

[3] Here the $\partial\chi/\partial x$ and $\partial\chi/\partial y$ are known by the Cauchy-Riemann conditions: essentially, they are like $\partial\psi/\partial x$ and $\partial\psi/\partial y$, but rotated counter-clockwise over 90°. Integration begins at an arbitrary "Greenwich" point in a central region and then works outward like an inkpot flooding algorithm in steps of ± 1 in either x or y direction. The boundaries of the panel act as a wall for the flooding.

Next the motifs are mapped. Each grid cell gets one motif. Inside each grid cell we do a kind of bilinear interpolation (bilinear interpolation is well-known, yet our adaptation to Béziers [5] is new to the best of my knowledge, see the last section of this supplement). The motif must be described in Bézier curves as well, and the curves are morphed by mapping their control points (Figure 4(a), also showing the user interface of our software[4]). Now we can forget the pot- and con-arrays whereas all the Béziers thus produced written to file in .svg format (Figure 4(b)). The .svg file can be opened and edited further in Adobe Illustrator or Inkscape, as shown in Figure 4(c), where one of the letters L of the double-L motif is being fine-tuned.

Computing Potentials from Boundary conditions

The theory of complex variables is closely related to the theory of 2D electric fields and from the latter we borrow concepts such as potentials and field lines. For a given panel pattern, we interpret its curved sides as conductors of fixed voltage or as "insulators" (at least, that's how I call them). Therefore, we can obtain the field lines by numerically solving am equation in a pixel grid with chosen boundary conditions. The field lines and equipotential lines, reinterpreted as Bézier curves [5] provide us a smooth grid for morphing the motif.

A potential ψ is determined by its boundary conditions, which can be of several types: Dirichlet conditions define ψ at certain places to have a fixed value; Neumann conditions define ψ to have a fixed derivative. There are more types of possible boundary conditions than just Dirichlet and Neumann conditions (e.g. Robin boundary conditions), the conditions used in the paper are "mixed" (some Dirichlet and some Neumann boundaries). In an application with four sides, *A*, *B*, *C*, and *D* say (*A* opposite *C*), we apply Dirichlet conditions to *A* and *C*, Neumann conditions to the other sides *B* and *D*. The Dirichlet conditions guarantee that *A* and *C* behave as capacitor plates (we set their voltage always to 1 and 0 volt), whereas the Neumann conditions guarantee that the field lines do not cross the side edges *B* and *D*. Configurations with more sides are also possible.

In very simple configurations, Laplace's equation can be solved by a formula (like $e^{\omega z}$ for the cone described in the first section of the main paper). In most applications however, these solutions are not available, and we must find approximations by numerical methods. One such method is the Schwarz–Christoffel mapping [1], but that only works for polygons (no known algorithms for Béziers, except algorithms of the relaxation type, as we use here). We use a method called Laplacian relaxation [4]. The area of interest in the complex plane is represented by an array, indexed by integers $0..j_{\max}-1$ and $0..i_{\max}-1$. The array is initialized at boundaries: the values at Dirichlet boundaries are frozen. A Bézier curve [5] is given by $z(t) = a_1(1-t)^3 + 3c_1(1-t)^2t + 3c_2(1-t)t^2 + a_2t^3$ for $t \in [0,1]$ for anchor points a_1, a_2, control points, c_1, c_2. Rounding the real and imaginary parts of $z(t)$ to integers allows us to assign and freeze[5] the voltages of the Dirichlet condition into the pot-array v (line thickness 2: the situation is like writing a line with the pencil tool in Microsoft Paint where pixelation is unavoidable and one must choose a discrete line thickness). The "relaxation" assignment $v[j,i] \leftarrow \text{avg}(v[j \pm 1, i \pm 1])$, that is,

$$v[j,i] \leftarrow (v[j+1,i] + v[j-1,i] + v[j,i+1] + v[j,i-1]) / 4$$

is repeated many times for all j in the range $0..j_{\max}-1$ and i in $0..i_{\max}-1$ such that (i,j) is not a boundary point. The Dirichlet (conductor) positions remain untouched. Handling the Neumann conditions was more challenging; inspiration came from **`github.com/yohanyee/laplace-relaxation`**. The boundary-position step is $v[j,i] \leftarrow \text{avg}(v[j \pm 1, i \pm 1])$ but now the averaging is only over free neighbors, i.e., the set of index pairs from $(j \pm 1, i \pm 1)$ that are not subject to a boundary condition.

---

[4] The code is available on **`github.com/LoeFeijs/HolomorphicMappings`**, but it is still very experimental.

[5] By "freeze" we mean that they are not to be changed by the assignment statement of the relaxation algorithm.

We compute the conjugate harmonic of the potential by path integration, starting from an arbitrary "Greenwich" point (j_0, i_0) which we assign the value 0. The paths consist of one-pixel steps following an inkpot-like flooding strategy[6]. The inkpot-like flooding strategy is chosen to avoid integrating closely along the edges, where the field is not accurate enough.

Then we find the field lines by steepest descent in the potential (pot) array and the equipotential lines by steepest descent in the harmonic conjugate (con) array. The spacing between the equipotential lines is straightforward now: if we want 11 lines, for example, we divide an arbitrary scaffold line going from 0 volt to 1 volt into $N - 1 = 10$ segments, locating the points of 0.1, 0.2, etc. volts. These are the starting points for the steepest descent procedure.

Interpolating inside Grid Cells defined by Bézier Curves

We sketch our bilinear interpolation between the Bézier curves of a grid cell in Figure 5. For the theory of Bézier curves, see [5]. Each control point or anchor point of the motif $z = x + iy$ is mapped into the grid cell indexed $j = \lfloor y \rfloor$, $i = \lfloor x \rfloor$ so *within* that cell we have interpolation parameters $t_x = x - \lfloor x \rfloor$ and $t_y = y - \lfloor y \rfloor$.

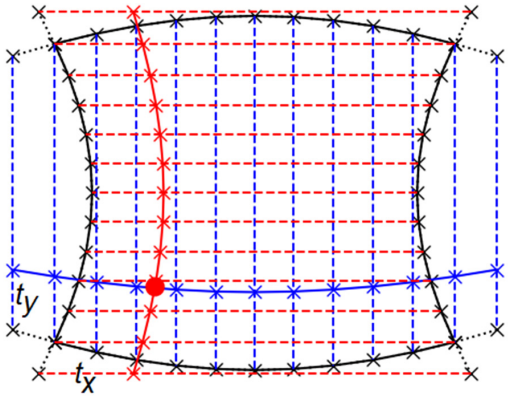


Figure 5: *Bilinear interpolation in grid cell.*

This, along with all other algorithms involved, is coded in Python 3.9 (**`www.python.org`**). Numpy (**`https://numpy.org`**) is used for working with large arrays and Matplotlib (**`https://matplotlib.org`**) for interactivity and displaying intermediate results. The Numba (**`https://numba.pydata.org/`**) just-in-time compiler for Python is indispensable for speeding up the computations.

Acknowledgements: We thank Troy Nachtigall and the Fashion Tech Farm and TU/e Wearable Senses communities for their en support and the Bridges reviewers for their valuable feedback.

References

[1] R.V. Churchill, J.W. Brown, and R.F. Verhey. *Complex variables and applications*, *third edition*. McGraw-Hill Kogakusha, 1974.

[2] N. Jonassen. *Electrostatics, second edition.* Kluwer 2002.

[3] Rosenbloom, P. (1956). "The method of steepest descent." *Proc Symp Appl Math*, Vol. 6, pp. 127–176).

[4] G.Steele. Partial Differential Equations 1: Boundary Value Problems. In: TN2513 Computational Science 2019/2020 (chapter 13). Online: https://nsweb.tn.tudelft.nl/~gsteele/TN2513_2019_2020/13%20Partial%20Differential%20Equations%201%20-%20Boundary%20Value%20Problems.html

[5] Wikipedia. Bézier curve. Online: https://en.wikipedia.org/wiki/B%C3%A9zier_curve

[6] https://www.freecodecamp.org/news/flood-fill-algorithm-explained-with-examples/

EXTRA MATERIAL

We add some images of the generated panels, such as for the “wave dress” and the “narrow-panels dress”.