EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

Sketch-Based Design for Bargello Quilts

M. Coahran† and E. Fiume

Department of Computer Science, University of Toronto

Abstract

Quilting is an art form in which amateur craftspeople explore geometry and color to create stunning designs. Bargello is a specific quilt style defined by a clever construction method that imposes constraints on the designs and gives them a characteristic appearance. Currently, Bargello patterns are typically designed manually, in a process that is laborious and time-consuming. We have developed a prototype system in which users can design Bargello quilts quickly and easily by sketching curves with a mouse. As a curve is drawn, the system transforms it into a graceful Bargello curve, composed of corner-connected axis-aligned rectangles, that respects both physical constraints and design constraints intrinsic to the Bargello style. We also provide a design setting that supports design variations typical of Bargello quilts and a set of tools to extend simple designs into complex ones. We conducted an informal evaluation of the system in a series of focus group sessions with potential users from the quilting community. Quilters were enthusiastic about the system and also provided suggestions for improvement.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Applications—Bargello quilt design I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation J.5 [Arts and Humanities]: Fine arts—Bargello quilts

1. Introduction

Quilting today is an art form in which amateur craftspeople explore compositions in geometry and color to create stunning designs. Bargello is a specific style within the world of art quilts defined by a clever construction method and characterized by parallel rows of gracefully flowing curves composed solely of rectangular pieces (Figure 1). Bargello patterns are typically designed via a laborious manual process. We have developed a prototype computer-assisted design system in which users can create Bargello quilt designs quickly and easily by sketching curves with a mouse.

1.1. Bargello quilts

The method quilters use to construct Bargello quilts is extremely efficient: a quilt may consist of hundreds of individual pieces, yet be produced with relatively few seams. Further, the method results in seams that are straighter, and a finished appearance that is crisper, than would be possible with a naïve construction method. However, the process imposes several constraints on Bargello designs. Therefore, it is necessary to understand the construction process in order to understand the Bargello design constraints.

Figure 1: *Traditional Bargello design, from [Edi94] page 40, used with permission of the author and publisher.*

To construct a Bargello quilt, one begins by cutting long

† mcoahran,elf@dgp.toronto.edu

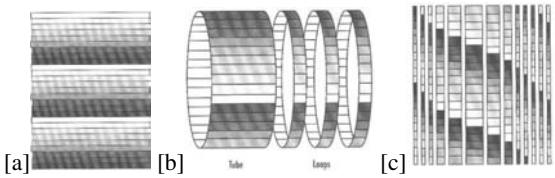

Figure 2: *Bargello construction process, from [Edi94] pages 5,7, used with permission of the author and publisher.*

narrow strips, called *color strips* from each of several fabrics [Edi94, Wil01]. The color strips are then stitched together lengthwise, into *color runs* or *color strata* (Figure 2[a]). Frequently, three or four color runs are stitched together to create a repeating pattern, and then the bottom color strip is stitched to the top color strip to form a tube (Figure 2[b]). Next, patchwork loops of various widths are created by cutting perpendicular slices off the tube. Finally, each loop is opened out into a linear strip, called a *vertical strip*, by removing the stitching in one seam or by cutting one fabric piece in two (Figure 2[c]). Consecutive loops are opened at consecutive fabric positions to create vertical strips that exhibit different phases of the same repeating pattern. The vertical strips may now be sewn together lengthwise to complete a quilt top that consists of a single set of parallel curves. Alternatively, the design can be *embellished* by modifying the vertical strips before stitching them together. For example, fabric sequences within a vertical strip can be reversed or replaced by other sequences to create more complex designs.

The design constraints imposed by this construction method are: all fabric pieces are rectangular, all are aligned in vertical columns, and the pieces that comprise a given color strip all have the same height. In addition, although not required for construction, it is common that all pieces in the entire design have the same height and that the pieces in adjacent vertical strips are either aligned horizontally to form rows, as in Figure 3, or offset by one half the height of a color strip, as in Figure 1. (These conditions are known as *matched seams* and *staggered seams*, respectively.) As a result, there is frequently only one geometric degree of freedom in a Bargello design: the width of the vertical strips.

Despite these constraints, Bargello designs can become extremely complex (Figure 3). Each color strip forms a curve through the design. The curve tangent as it passes through a given fabric piece is defined by the height and width of the piece. Since all pieces have the same height and the pieces in a given vertical strip have the same width, each curve as it passes through a vertical strip is either parallel or symmetric to every other curve that passes through the same strip. This can result in designs that consist entirely of parallel curves or in vertically symmetric designs. However, the curves need only be locally parallel or symmetric. If the extrema in two curves are not aligned horizontally, then the curves will be neither globally parallel nor globally symmetric. This allows

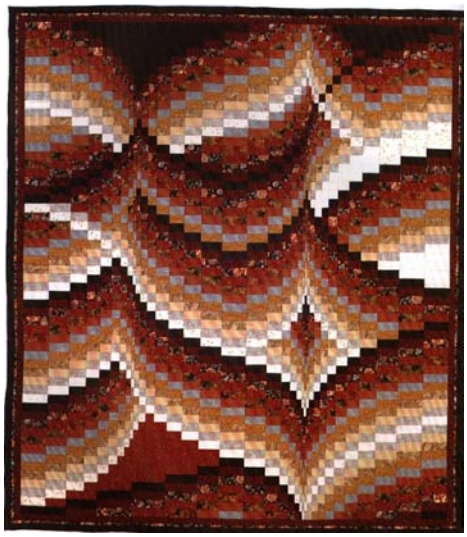

Figure 3: *Contemporary Bargello design, from [Edi94] page 72, used with permission of the author and publisher.*

Bargello designs to become arbitrarily complex with color strips that run parallel to one another in some regions and symmetric to one another in others.

Typically, Bargello curves are designed manually. First, the desired curve is sketched on graph paper, and then in another location an appropriate set of grid squares is penciled in to represent the curve [Edi94]. This is more difficult than it sounds. Graph paper is much lower resolution than the sketch, so a naïve "scan conversion" of the curve to the grid can result in an unattractive rasterization. If the apex of a curve does not fall at an optimal phase of the grid, an unmodified scan-conversion algorithm can result in an overly wide or an overly narrow tile at the apex. Further, the sketch may include a segment that is too steep to be represented by a set of corner-connected tiles. Therefore, an accomplished designer will create a stylistic abstraction of the sketch, not a mechanical scan conversion.

We are aware of two commercial software packages for creating Bargello designs [Bar,Ele], but the support they provide for the design process is rudimentary. In our experience, it is simplest to design a curve on paper and enter the completed geometry into the program.

1.2. Sketch-based design

Sketch-based design systems allow users to create designs via freeform sketches, bypassing the laborious manipulation of individual control points required by traditional CAD systems. The resulting designs lose the high precision offered by CAD, but in many settings high precision is not required and the artistic beneift can be substantial. Designs can be created quickly, and designers can focus on the creative aspects of the design process. Further, sketch-based design

systems are easy to learn and accessible to beginning designers. However, the less information the designer articulates, the more the system needs to infer. Research in this field has explored various kinds of inferences that can be drawn in various contexts.

Some sketch-based systems have focused on recognizing low-level primitives and the relationships between them. These systems do not presume particular application domains, but they do make assumptions about attributes that are desirable in drawings. Pavlidis and Van Wyk [PW85] developed a system that "beautified" line drawings by adjusting the lines to make approximate relationships between them precise. Pegasus [IMKT97] expanded on this by recognizing an extended set of relationships and by suggesting multiple beautified line placements when conflicting relationships were discovered. Fluid Sketches [AN00] continuously examined strokes as they were being drawn, classifying them as lines, circles, or boxes, and morphing the sketch toward the best-fit instance each time an input point was received.

Other sketch-based systems have explored the creation of 3D objects based on 2D input strokes. To manage the inherent ambiguities in the problem, these systems must limit the domain of shapes they can create. *SKETCH* [ZHH96] allowed users to create and manipulate geometric solids with gestural commands. *Teddy* [IMT99] created 3D characters from freeform sketches by interpreting closed strokes as silhouettes and inferring 3D geometry from them. To make this possible, Teddy made a strong assumption about the objects in its domain, and with those assumptions all objects acquired a similar "rotund" morphology.

Still other systems have focused on domain-specific early-stage design. These systems intentionally retain the "sketchy" appearance of the original input strokes to reflect the amorphous nature of early-stage designs. The domain specificity of SILK [LM95], a user interface design system, allowed it to recognize high-level objects such as scroll bars and buttons. The Electronic Cocktail Napkin [GD96] extended this by allowing users to specify new application domains and teach the system stroke configurations relevant to each domain.

Our system applies a sketch-based design approach to a new application domain. Similar to [IMT99], all input curves are known *a priori* to belong to the specialized class of Bargello curves, and our task is to create a 2D Bargello virtual fabric tiling from the best-fit instance. Like [AN00], we continuously update the resulting tiling as the sketch is being drawn. Like [LM95], our focus is on creating real designs in a specific application domain, and we encounter and respect the constraints of that domain.

1.3. Contributions

We present an algorithm that transforms sketched input data into graceful Bargello curves, consisting of a set of corner-connected axis-aligned rectangle primitives, that respect both the physical and the design constraints of the Bargello style. The algorithm incorporates novel curve fitting, rasterization, and beautification schemes specific to Bargello curves. Further, the algorithm is extended to the case of multiple independent curves that share the same columnar region within a Bargello design; this problem is more difficult since the independent curves must be coordinated to create a single set of design columns.

In addition, we present a sketch-based Bargello quilt design system in which users can create Bargello designs quickly and easily. The system addresses an authentic need, given that the current manual design process for Bargello quilts is laborious and time-consuming.

2. Bargello curve generation

We define a *Bargello curve* as a connected path of rectangular tiles, all rendered with the same fabric texture, within a Bargello design. Our objective is to generate a "graceful" Bargello curve that approximates a sketch drawn with a mouse or stylus, while respecting the constraints of the domain. Our approach is to generate a piecewise Hermite cubic curve that approximates the input data, rasterize the Hermite curve to produce an initial tile path, and then beautify the tile path.

2.1. Curve fitting

Bargello designs are typically composed of twenty to sixty rows of fabric tiles, so they are inherently low resolution. Thus, Bargello curves tend to be large sweeping curves (with respect to the design space) with various local extrema in wide arcs and narrow peaks. We have found that if an arc between inflection points in the Bargello curve is spanned by multiple Hermite segments, the resulting rasterization tends to look ragged. In such a case, the underlying Hermite segments model high-frequency noise with respect to the Bargello curve. Therefore, we generate piecewise Hermite curves that match the frequency of the Bargello curve by placing knots at each of the local extrema and inflection points of the Bargello curve. Thus, each Hermite segment is monotonic in both x and y.

First, the input data, which are originally sampled at approximately 24-40 Hz, are thinned such that each remaining point is at least $\frac{3h}{4}$ from its neighbors, where h is the row height in the current Bargello grid. This reduces the amount of subsequent computation and retains sufficiently many points to specify the curve at the desired resolution.

Next, the data are scanned to find points that represent local extrema and inflection points of the curve. We seek extrema in x and y because the peaks and valleys characteristic of Bargello curves are of necessity aligned with the vertical columns of the Bargello design, and Bargello curves may

contain switchbacks that create horizontal extrema. However, they cannot contain cusps at other angles without rotating the entire design. The data is scanned in three passes, one each to detect x-extrema, y-extrema, and inflection points between them. The separate passes allow "false extrema" to be detected and discarded immediately.

The first pass seeks local x-extrema. Let P_{x_i} be the x coordinate of data point P_i. Let σ_i be defined as

$$\sigma_i = (P_{x_{i+1}} - P_{x_i})(P_{x_i} - P_{x_{i-1}}). \tag{1}$$

For each P_i, if σ_i is negative, P_i is recorded as an x extremum. However, if the distance in x between P_i and the previously recorded x extremum is below a threshold (e.g., $\frac{3h}{4}$), P_i is not accepted as an extremum, and the previously recorded extremum is discarded. This ensures that the accepted extrema represent sustained direction changes, rather than noise with respect to the low resolution Bargello design. The second pass scans the data between adjacent x-extrema, seeking local extrema in y. It is analogous to the first pass, with x's replaced by y's in Eq. 1.

The third pass seeks inflection points between any of the discovered extrema. For each P_i between adjacent extrema, a bounded proxy b_i for the slope of line segment P_iP_{i+1} is computed according to Eq. 2, where K is either $+1$ or -1. A proxy b_{i-1} for line segment $P_{i-1}P_i$ is computed analogously:

$$b_i = K \, \frac{(P_{y_{i+1}} - P_{y_i})^2}{(P_{x_{i+1}} - P_{x_i})^2 + (P_{y_{i+1}} - P_{y_i})^2} \, . \tag{2}$$

Then a measure of the discrete curvature c_i is given by $c_i = b_i - b_{i-1}$. The resulting c_i are passed through a three-point moving average filter since noise in the input data is exaggerated by discrete curvatures. Finally, the filtered c_i are scanned for sign changes. When a sign change is detected, the associated data point is recorded as an inflection point of the curve, with a similar caveat to that used for extrema. If the new inflection point is too close to the previous one, both are discarded. For this test, we have found that a distance threshold of approximately h works well.

Finally, cubic Hermite curve segments are generated between adjacent knots via least-squares approximation. In this calculation, the position of each knot is constrained, and we solve for the associated tangents. C^1 continuity is not enforced because Bargello curves frequently contain cusps where tangent discontinuities should be retained. Further, the Hermite curve is never displayed, and the fabric tiles of the Bargello curve obscure minor tangent discontinuities at non-cusp knots.

2.2. Rasterization and beautification

The next task is to convert the piecewise Hermite into a connected path of rectangular tiles. In addition to reflecting the shape of the Hermite curve, the Bargello curve should conform to the physical and design constraints of the domain. Specifically, a minimum tile width should be imposed, and only discrete tile width increments should be allowed. These requirements arise because fabric pieces will be cut with discretely marked rulers and must be large enough to be manipulated by hand. To conform to the Bargello construction process, all tiles in the design should be aligned in columns, and the tiles within a given Bargello curve should be the same height. Further, consecutive tiles should occupy different design columns, i.e., there should be no vertically adjacent tiles in a Bargello curve. The latter constraint implies that the rasterization should not contain any "elbows."

Finally, the Bargello curve should be "graceful." The requirements for a graceful curve are less clear, but we have defined the following soft constraints in pursuit of this goal. First, on each arc between inflection points, the constituent tiles should exhibit either a monotonically increasing or a monotonically decreasing sequence of tile widths, allowing repeated widths. Second, no tile should be more than double the width of adjacent tiles, except when an adjacent tile is the minimum allowed width. The neighbors of a minimum width tile may be as much as three times the minimum tile width. Third, in a design that includes multiple independent Bargello curves, tiles in one curve may need to be subdivided into multiple design columns to accommodate the tangent requirements of another curve. In this case, subdivisions that result in narrow tiles should be avoided.

DATA STRUCTURES. A Bargello design is stored in a "bitmap" of non-square cells. We refer to the geometry underlying the bitmap as the *Bargello grid* and reserve the term bitmap to refer to the data structure itself. A fabric tile will occupy either one or two rows, for "matched seams" or "staggered seams" designs, respectively. However, a tile may occupy several columns in the bitmap since each grid column represents the finest allowable tile width increment. The minimum tile width can be set to any positive integer multiple of the minimum width increment.

The output of a scan-conversion algorithm, described next, is stored in a data structure called a *tile path*. This tile path is used to initialize a beautification process that generates a second tile path, and the second tile path is painted into the bitmap with a z-buffer algorithm. Both tile paths are stored in a data structure called a *Bargello curve*. Storing the beautified tile path allows it to be written into the bitmap repeatedly, without repeated beautification. However, sometimes re-beautification is necessary, so the original tile path is retained for subsequent initializations.

ALGORITHMS. Bargello curves suffer from some of the same scan-conversion hazards text characters do. If an extremum of the piecewise Hermite lies close to a row boundary in the Bargello grid, the resulting rasterization will contain either a long flat tile or an isolated thin tile at that point. Neither outcome reflects the shape of the underlying curve well. A uniform translation of the curve does not repair the problem since various extrema fall at various phases

of the grid. Therefore, following an approach used in the rasterization of text characters [Her87], the Hermite curve is stretched to place each y-maximum and y-minimum at the nearest point that lies $\frac{3}{10}h$ and $\frac{7}{10}h$, respectively, from the bottom of a grid row, where h is the grid row height. Although extrema may be moved to adjacent rows, all moves are "subpixel" with respect to the Bargello grid. Further, there is no danger of moving a local maximum to a position below an adjacent local minimum because such a pair would have been discarded as "false extrema" during the curve-fitting process.

Next, a rasterization of the stretched curve is computed and stored in a tile path. To ensure that the tile path is connected, each Hermite segment is traversed with a uniform step size δt that ensures no grid cell is missed. The step size is defined as

$$\delta t = \min\left(\frac{w}{\max_{0\le t\le 1}|\frac{dx}{dt}|}, \frac{h}{\max_{0\le t\le 1}|\frac{dy}{dt}|}\right), \qquad (3)$$

where each of the maxima are taken over the curve segment, and w and h are the width and height of each grid cell, respectively. This choice of step size is based on a result by Kaufman [Kau87] but has been adapted to accommodate rectangular grid cells.

After computing the initial tile path, a copy is made, and the copy undergoes beautification. Two beautification algorithms are used: a greedy algorithm that runs while the curve is being drawn, and a dynamic programming algorithm that runs when the pen is lifted.

Greedy beautification is performed as the curve is drawn because each new data point can influence the entire previous curve segment. However, only a subset of the design constraints are enforced at this time. First "elbows" are removed, and then the tile path is massaged until it meets the monotonicity constraint. The tiles on each arc must exhibit either a monotonically increasing or decreasing sequence of widths. Because each arc is monotonic in both x and y, the appropriate sequence direction can be determined by the tangents at the endpoints of the underlying Hermite segment. The tile path is traversed and grid cells are "pushed" from one tile to the next if they do not meet the constraint.

The second beautification algorithm seeks a globally optimal solution via dynamic programming and considers all the constraints. The recursive cost definition is given by

$$E_{i,j} = \min[E_{i-1,k} + e_{k,j}], \text{ over all feasible } k, \qquad (4)$$

where $E_{i,j}$ is the minimum cost for any tile path up to and including tile i such that tile i has width j, and $e_{k,j}$ is the cost of a tile having width j given that the previous tile had width k. Details of $e_{k,j}$ will be given presently.

The algorithm computes a sequence of tile widths, stated as the number of grid cells occupied, but the number of tiles and their vertical positions are taken directly from the reference tile path. The solution includes an invisible tile that extends from the left edge of the design space to the leading edge of the first tile, and this tile is allowed to have zero, positive, or negative width. (The latter case can occur if the tile path is dragged partially off the design space after the curve is drawn.)

The first grid cell occupied by a given tile is always horizontally adjacent to the last cell occupied by the previous tile, so the tile path does not contain rasterization "elbows." In addition, several constraints are enforced by disallowing infeasible options. First, tiles that fall short of the minimum width are disallowed, and except for the last tile in the tile path, tiles that end less than the minimum width away from the edge of the design space are also disallowed since they render the subsequent tile infeasible. Second, while a user may drag a tile path partially off the design space, no tile is allowed to straddle the edge. This ensures that every tile is either completely visible or completely invisible, and therefore also ensures that the computed cost $e_{k,j}$ is correct for the visible portion of the tile. Finally, if a tile in the reference tile path abuts the edge of the design space, some tile in the beautified tile path must as well. This ensures that a tile path intended to cover the design space does so.

Four remaining soft constraints give the tile path a graceful appearance and retain the shape of the underlying Hermite curve. These objectives are blended together as follows:

$$e_{k,j} = w_d c_d + w_m c_m + w_r c_r + w_s c_s \; . \qquad (5)$$

We now motivate and define c_d, c_m, c_r and c_s.

As tile i is placed in the tile path, the position of its leading edge has already been determined, and we seek the best position for its trailing edge. The algorithm works equally well for curves drawn right to left or left to right. The *distance cost*, which keeps the tile path close to the underlying Hermite curve, is defined as

$$c_d = \left|x_{r_i} - x_{b_i}\right| . \qquad (6)$$

Here, x_{r_i} and x_{b_i} indicate the horizontal position of the trailing edge of tile i in the reference and beautified tile paths, respectively.

The *monotonicity cost*, c_m, encourages the tile path to exhibit monotonic tile width sequences between inflection points:

$$c_m = \left|\frac{j-k}{j+k}\right| . \qquad (7)$$

This cost is imposed if tile i is wider than tile $i-1$ when it should be narrower, or vice-versa; otherwise, $c_m = 0$. In this equation, j and k are the widths of tiles i and $i-1$, respectively. The cost c_m is greater in steep regions of the tile path than in shallow regions because a lack of monotonicity is more apparent between narrow tiles.

The *width ratio cost*, c_r, encourages the tile path to exhibit

smooth width transitions between adjacent tiles:

$$c_r = \max\left\{\frac{j}{k}, \frac{k}{j}\right\}. \tag{8}$$

Here, j and k retain their previous definitions, and c_r increases as the difference between them increases. The cost is imposed if $\max\left[\frac{j}{k}, \frac{k}{j}\right] > 2$; otherwise, $c_r = 0$. An exception is made when either j or k is a minimum width tile. To keep the curve from losing flexibility in steep regions, tiles adjacent to minimum width tiles are allowed three times the minimum width without penalty.

These three cost terms relate to the current Bargello curve alone. If it is the only curve in the design space, they are sufficient, but when multiple Bargello curves share the design space a *subdivision cost*, c_s, is introduced, which we now discuss.

After beautification, a Bargello curve is "painted" into the bitmap. Since all tiles in the design must be aligned in columns, the vertical edges between adjacent tiles in the tile path imply boundaries between vertical columns that run the full height of the design space. If a Bargello curve is drawn when the design space already contains a curve, the space has already been partitioned into columns. In all likelihood, the tangents of the new curve will differ from those of the pre-existing one, and the tiles in the new tile path will not be aligned with pre-existing column boundaries. Then new column boundaries are inserted into the design space, subdividing the pre-existing columns. The data structure of the pre-existing curve is not modified, but the corresponding tiles are subdivided *de facto* in the design, and they will be constructed from separate fabric pieces in the physical quilt.

This mechanism allows multiple independent curves to co-exist in a design. However, arbitrary subdivisions are unattractive and unnecessarily complicate quilt construction. Optimally, we would like narrow tiles from one tile path to fit snugly within a wider tile from the other tile path. In contrast, when tiles from two tile paths interlock like brickwork the subdivisions are unnecessary, especially if the resulting design columns are narrow. It would be preferable to collapse the columns and bring the curves into alignment.

Therefore, we impose a *subdivision cost* on tile paths that subdivide existing columns in the design space. This cost discourages brickwork-style subdivisions, especially when the resulting columns are narrow, but it does not discourage subdivisions that are completely internal to an existing column since these may contribute to a snug-fit set of subdivisions.

Let x_0 and x_1 be the horizontal positions of the left and right edges of tile i, respectively. Let x_a through x_d be the horizontal positions of column boundaries that already exist in the design space, where x_a indicates the rightmost boundary such that $x_a \leq x_0$, x_b indicates the leftmost boundary such that $x_0 < x_b$, x_c indicates the rightmost boundary such that $x_c \leq x_1$, and x_d indicates the leftmost boundary such that $x_1 < x_d$. Then, by definition, $x_a \leq x_0 < x_1 < x_d$. However, there are several scenarios for the placement of x_b and x_c, some of which are shown in Figure 4.

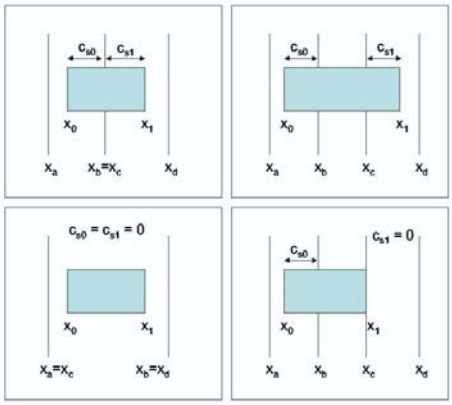

Figure 4: *Design column subdivisions.*

If x_0 lands inside the design column delimited by x_a and x_b, and x_1 lands outside it (i.e., $x_a < x_0 < x_b < x_1$), then tile i straddles the right edge of the column, and cost c_{s_0} in Eq. 9 imposes a penalty inversely proportional to the distance between x_0 and x_b. In contrast, if x_1 lands within the same design column (i.e., $x_1 < x_b$), then the column is partitioned in at least three pieces, and no cost is imposed on tile i. If a subsequent tile straddles the boundary, cost c_{s_0} will be imposed on the subsequent tile.

$$c_{s_0} = \begin{cases} \frac{1}{x_b - x_0} & \text{for } x_a \neq x_0 \text{ and } x_b \leq x_1. \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Similarly, if x_1 lands within an existing design column and x_0 lands outside it (i.e., $x_0 < x_c < x_1 < x_d$), then tile i straddles the left edge of the design column, and cost c_{s_1} in Eq. 10 imposes a penalty inversely proportional to the distance between x_c and x_1. Again, if x_0 lands within the same design column (i.e., $x_c < x_0$), no cost is imposed on tile i.

$$c_{s_1} = \begin{cases} \frac{1}{x_1 - x_c} & \text{for } x_0 < x_c < x_1. \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Finally, the subdivision cost c_s is given by $c_s = c_{s_0} + c_{s_1}$.

The weights in Eq. 5 can be varied to modify the balance between the objectives. We use $w_d = 0.5$, $w_m = 30$, $w_r = 5$, and vary w_s from 0 to 4, as will be explained next. These settings result in monotonicity and width ratio costs that are typically approximately equal, while the subdivision cost is somewhat less, and the distance cost is less still. This reflects our preference for Bargello curves that are graceful over ones that follow the sketch exactly, given that sketches are likely to be intended as evocative of the desired result. Similarly, while we strive to reduce the number of tile subdivisions, we do not do so at the expense of gracefulness.

In this algorithm, the current tile path adjusts itself to

align with pre-existing columns in the design space, but pre-existing tile paths do not reciprocate. It would be preferable if all the tile paths adjusted themselves to one another simultaneously. However, a dynamic programming algorithm to find a globally optimal fit among an arbitrary number of tile paths would require a matrix of arbitrary dimensions! Thus, we take a different approach. A high-level multi-pass algorithm cycles through the tile paths and invokes the dynamic programming algorithm to fit each tile path in turn to the design columns defined by the others. On the first cycle, the subdivision weight w_s is set to zero, so each tile path finds its own ideal shape. Thereafter, on each cycle w_s is increased, and the tile paths move into alignment with one another.

There is one more case to consider. A Bargello curve that is not single-valued in x presents a challenge similar to that presented by multiple independent curves. Various segments of the curve share columnar regions within the design space, and assuming their tangents differ, they each insert column boundaries that subdivide one another's tiles. However, the algorithm as described so far has no knowledge of design columns defined by other segments of the same curve. To rectify this, curves that are not single-valued in x are parsed into segments between x-extrema. On each pass of the high-level algorithm, each segment is fit to the design columns defined by the others. Tiles in each segment are aligned with tiles in the others, but the segment endpoints may move with respect to one another, disconnecting the tile path. Therefore, after each segment has been fit individually, the full curve is fit to the design columns established by the individual segments, so the final beautified tile path follows the beautified segment templates to the extent possible while maintaining connectivity.

The beautification algorithm produces graceful Bargello curves that approximate the input sketches well in many cases. However, it is possible to draw a curve that is too steep to follow with discrete corner-connected tiles. In response, the algorithm produces a linear segment of minimum width tiles. If the segment is not long, the resulting curve still looks graceful, but long linear segments can be unattractive and may not approximate the underlying sketch well. Possible solutions include displaying guidelines while users draw that indicate whether a given curve can be followed, and allowing tile path connectivity to be broken in order to retain the curvature of the sketch. This may also surprise users, but it could offer a visual explanation of the underlying difficulty.

3. Computer-assisted Bargello design

We have developed a prototype computer-assisted design system for Bargello quilts. It consists of two modules: one for selecting fabric images, and the other for creating Bargello quilt designs. The problems addressed by the system include some specific to Bargello and others endemic to quilting in general. For one, the current design process for Bargello quilt patterns is tedious. For another, finished quilts frequently look different than the designer had imagined, which can be disappointing given the amount of time and money expended. This typically results from using fabrics that appear differently in combination than they do individually, given that colors are perceived differently depending on the context in which they are viewed. Our goal is to help alleviate these problems by allowing quilters to create and experiment with designs quickly and to visualize the designs with real fabric textures so that disappointing designs can be discarded painlessly, and only the most promising will be brought to fruition in fabric.

3.1. Fabric selection module

The fabric selection module provides access to a database of real fabric images and contains various widgets that allow users to experiment with color combinations and fabric orderings in Bargello designs (Figure 5 and Color Plate).

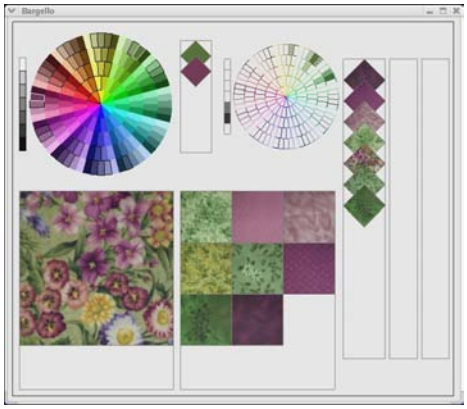

Figure 5: *Fabric selection module.*

The *main color wheel*, in the upper left corner, is used to drive color selection. Black outlines around some color tiles indicate the presence of fabrics that contain those colors and are not currently displayed. Clicking on a color tile adds that color to the current "color scheme." In response, the selected tile is outlined in white, a larger tile in the same color is added to the *color scheme palette* just right of the main color wheel, and fabrics containing that color are added to the *fabric browser* at the bottom of the screen.

The fabric browser includes two windows: fabrics displayed in the right, or left, window contain significant amounts of one, or more than one, of the color scheme colors, respectively. Clicking on a fabric texture in either window selects it for a Bargello design and adds it to one of the three *fabric palettes* along the right edge of the screen.

At any given time one of the fabric palettes is "active," and one of the fabric tiles within that palette is also active. The active tile can be removed or dragged to a new position within the palette. When fabrics are selected they are added to the active palette, and when Bargello curves are drawn

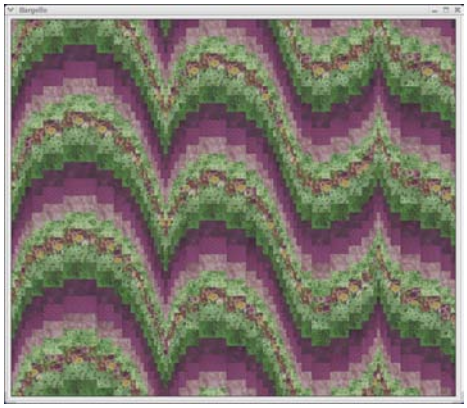

Figure 6: *Traditional, with staggered seams.*

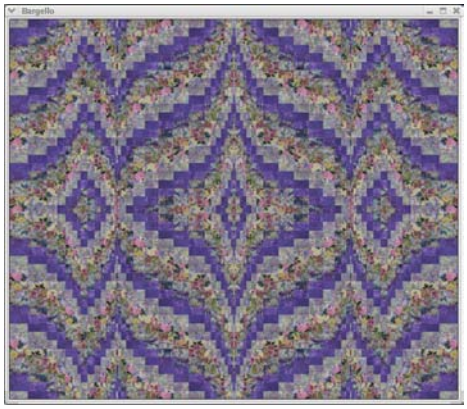

Figure 7: *Symmetrical, with matched seams.*

they are associated with the active palette. Thereafter, modifying the fabrics or the fabric ordering within the palette causes a corresponding modification in the Bargello design. Three palettes accommodate designs with up to three separate color strata. An unlimited number of palettes could be provided, but in practice we have found three to be sufficient.

Finally, the *color composition wheel*, left of the fabric palettes at the top of the screen, provides information about the colors and the amounts of each color in the currently active fabric. In addition, pop-up windows associated with individual color tiles allow users to browse and select fabrics containing colors that match those in the active fabric.

3.2. Bargello design module

A traditional design (Figure 6) can be created in the Bargello design module by drawing a curve with a mouse, starting near one margin and proceeding to the other while tracing out the desired inflections. As the user draws, the system transforms the input sketch into a graceful Bargello curve and spawns parallel curves to fill the vertical extent of the design space. If the stroke begins or ends sufficiently near the edge of the design space, the resulting Bargello curve snaps to the edge; however, if the stoke endpoints are well inside the design space, the Bargello curve is placed as indicated. Symmetrical designs (Figure 7 and Color Plate) can be created similarly by drawing in one of four pre-defined *reflection templates*.

Preferences can be set that affect system behavior in response to sketched input. For example, by default if a user reverses the horizontal direction of an input stroke, the latter portion of the Bargello curve is erased as the stroke backtracks over it. However, another setting allows users to draw curves with "switchbacks" (Figure 11). Similarly, by default the full vertical extent of the design space is filled as the Bargello curve is drawn, but another setting allows users to draw curves that consist of a single color strata (Figures 10, 11, 12). In addition, users can select a set of color strips within an existing design, keep those strips, and discard the rest. Thus, an arbitrary number of color strips can be associated with a given Bargello curve.

Once an initial curve has been drawn, additional curves can be added to a design in three ways. First, multiple independent curves can be drawn with the mouse or stylus (Figure 12 and Color Plate). Second, existing curves can be selected, and exact or reflected copies of them can be created (Figure 10). Third, horizontal stripes can be added to fill background regions of a design (Figure 9).

Users can also experiment with their designs in several ways. Curves can be reflected vertically or dragged. The fabrics or the fabric ordering in a color strata can be modified, and the fabric tiles can be arranged with matched or staggered seams. Further, the system supports a set of low-level design operations for creating complex contemporary designs. Users can delete a curve or tiles within a curve, alter the apparent depth of a curve or tiles within a curve with respect to other curves in the design, and substitute different fabrics into a set of tiles without modifying the default fabrics associated with the curve. To facilitate these operations, support is provided for selecting an entire curve or a set of tiles within a curve for subsequent modification.

At present, the various design operations and preference settings are invoked via keystroke commands. However, we envision a multi-modal interface in which curve geometry is specified by sketching, fabrics are selected via the custom widgets presented, and other operations are invoked with WIMP-style widgets. In our view, such an interface would serve the intended audience well. We anticipate that most quilters will be familiar with the WIMP paradigm, and they may prefer to focus on quilt design rather than on mastering new interface modalities.

4. Results

Figures 6 through 12 were created in our system.

Figure 8: *Arches.*

Figure 9: *Jewel.*

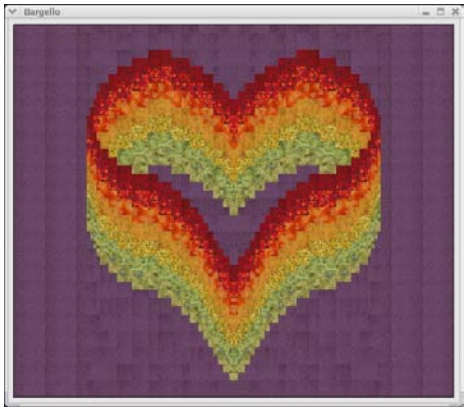

Figure 11: *Heart.*

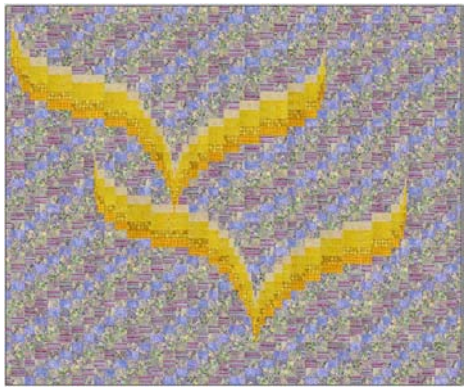

Figure 12: *Soaring.*

5. Feedback from quilters

Modern-day quilting guilds are social organizations of amateur quilting enthusiasts. A web search was conducted to identify guilds in the area, and electronic mail was sent to ten guilds describing the project and inviting members to participate in an evaluation of the system. Three meetings were arranged, and each meeting was attended by a small group of self-selected guild members. Twenty-one quilters participated in total. Each focus group session included time devoted to focus group style questions, a software demonstration, and an opportunity for participants to experiment with the system. In each group, participants asked questions and made helpful suggestions throughout the meeting.

Figure 10: *Weave*

Quilters were enthusiastic about the system. In particular, they were impressed with how quickly and easily Bargello curves could be designed. When the first curve was drawn in each demonstration, participants responded with exclamations of surprise: “I want your program! Wow!” and “What a feeling of power! Think how long that would take to design on graph paper.” In addition, they thought the ability to experiment with fabrics and to view designs rendered with real fabric textures would be helpful to quilters, and they told us the system was entertaining and fun to use.

Participants also offered suggestions for improvement. For example, they unanimously requested support for editing curves and for practical operations such as computing yardage requirements and printing instructions for quilt construction. In addition, they requested the ability to erase a portion of a curve by backtracking over it with the mouse while drawing. Some of their suggestions have been implemented, and others remain as future work.

6. Conclusion

We have developed a prototype system in which users can create Bargello quilt designs by sketching curves with a mouse. As the user draws, the system transforms the input sketch into a graceful Bargello curve consisting of corner-connected tiles that respect both physical and design constraints of the Bargello style. Support is provided for multiple independent curves and curves that are not single-valued to co-exist within a design. The system provides features that support a variety of design styles. Simple designs can be created with a single sweep of the mouse, reflection templates support designs with various symmetries, and a set of low-level operations is provided to extend simple designs into complex ones. The system also provides a module for making fabric selections and experimenting with color combinations and fabric orderings in a design. Thus, users can visualize their designs rendered in real fabric textures, reducing the possibility of disappointment in the completed physical quilt. We sought feedback on the system from user groups in the community. Quilters were enthusiastic about the system, especially how quickly and easily Bargello curves can be designed, and they also provided suggestions for improvement.

However, the system is a first-generation prototype, and there are many ways it could be improved and extended. For example, it currently does not provide support for editing existing Bargello curves. To refine the shape of a curve, users must clear the design and re-draw it. Given that sketching a new curve is quite easy, we have found this method workable in the short term: a curve can be drawn repeatedly and improved over time. However, the ability to modify existing curves would be a marked improvement.

In addition, although it is possible to create non-traditional Bargello designs in our system (e.g., Figures 9 through 12), it is not as easy as we would like. The system supports several operations intended for this purpose, but they fall short of providing the intuitive interface we had hoped for. More investigation is needed to determine the set of operations and features that could provide an intuitive way to create complex contemporary designs. Collaboration with experienced Bargello designers will be essential to the solution of this problem.

There are also several practical issues to address. For example, users should be able to compute yardage requirements and print detailed instructions for quilt construction. Focus group participants also requested the ability to scale existing designs to various standard sizes.

Finally, we envision extending this work to include other quilting styles. For example, to our knowledge no quilt design software supports the design of landscape quilts. Such a system could allow users to "paint" scenes using input strokes that are rendered with real fabric textures. Subsequently, the system could produce sewing patterns based on stroke geometry. Alternatively, the software could detect strokes in existing images and allow users to substitute fabric textures into each stroke in the scene. Conceivably, the system could solve for an optimal set of fabric textures without user intervention; however, this may not be desirable. We are wary of removing the artistic aspects of quilt design from the hands of human artists. Our goal is to provide computational tools that support human creativity, not to supplant it.

7. Acknowledgments

We thank the quilters who participated in our focus groups for sharing their time and their insights with us. We thank the Simcoe County Quilt Shoppe for allowing us to photograph fabrics from their inventory. Finally, we thank John Hancock, Michael Neff, and Gonzalo Ramos for their assistance with video production.

References

[AN00] ARVO J., NOVINS K.: Fluid sketches: Continuous recognition and morphing of simple hand-drawn sketches. *UIST '00* (2000), 73–80. 3

[Bar] Bargello designer. http://www.sheilawilliams.com/bargo/Bargello.html. 2

[Edi94] EDIE M.: *Bargello Quilts*. Martingale and Company, Woodinville, WA, 1994. 1, 2

[Ele] Electric quilt 5. http://www.electricquilt.com. 2

[GD96] GROSS M., DO E. Y.-L.: Ambiguous intentions: A paper-like interface for creative design. *UIST '96* (1996), 183–192. 3

[Her87] HERSCH R.: Character generation under grid contraints. *Computer Graphics 21*, 4 (1987), 243–252. 5

[IMKT97] IGARASHI T., MATSUOKA S., KAWACHIYA S., TANAKA H.: Interactive beautification: A technique for rapid geometric design. *UIST '97* (1997), 105–114. 3

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. *SIGGRAPH '99 Conference Proceedings* (1999), 409–416. 3

[Kau87] KAUFMAN A.: Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *Computer Graphics 21*, 4 (1987), 171–179. 5

[LM95] LANDAY J., MYERS B.: Interactive sketching for the early stages of user interface design. *CHI '95 ACM Press* (1995), 43–50. 3

[PW85] PAVLIDIS T., WYK C. V.: An automatic beautifier for drawings and illustrations. *Computer Graphics 19*, 3 (1985), 225–234. 3

[Wil01] WILLIAMS B.: *Colorwash Bargello Quilts*. Martingale and Company, Woodinville, WA, 2001. 2

[ZHH96] ZELEZNIK R., HERNDON K., HUGHES J.: Sketch: An interface for sketching 3D scenes. *Computer Graphics 30*, 4 (1996), 163–170. 3