Artificial Intelligence for Engineering Design, Analysis and Manufacturing

cambridge.org/aie

Shape computations with NURB curves

Elif Ensari and Mine Özkar

Istanbul Technical University, Faculty of Architecture, Taskisla, Taksim, Istanbul 34437

Abstract

Freeform curves are commonly used in contemporary design practices, especially with digital modeling tools. We investigate facilitating shape subtraction and addition with two-dimensional (planar) non-uniform rational basis-spline (NURB) curves with the codes and conventions of modeling while preserving the visual continuity of curved shapes. Our proposed tool, developed in a common digital modeling environment, automates the adjustment of parameters for tangential continuity of curves in shape rule applications. When the user designates a curve range to subtract from an initial shape and provides a new curved shape to add to it, the tool splits the initial shape, scales and aligns the curve to be added to fit into this range, introduces additional control points at the joining ends of the new curve to preserve continuity and redraws the new curve. We present a sample set of design variations produced using this practical approach which can be utilized as a method or become part of an automated NURB curve manipulation tool for designers.

Research Article

Cite this article: Ensari E, Özkar M (2018). Shape computations with NURB curves. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **32**, 282–294. https://doi.org/10.1017/S0890060417000592

Received: 25 October 2016
Revised: 21 August 2017
Accepted: 24 August 2017

Key words:
Computer implementation; curve implementation; NURB curve grammars; shape grammars

Author for correspondence:
Elif Ensari, E-mail: elif.ensari@gmail.com

Introduction

Non-uniform rational basis-spline (NURB) curves have come to be commonly used in computational design tools, especially since the introduction and the rapid advancement of NURB surface modeling software into design practice. Freeform curves, mostly modeled as NURB curves, are parts of many contemporary design vocabularies. Implementations of shape grammars have previously explored curves but there have been limitations in dealing with NURB curves. An implementation that incorporates NURB curves holds potential for integrating shape computation with conventional design tools and in generative design processes providing the designer a means of systematically working with multiple NURB curves.

In shape grammars, three primary steps of a shape rule application are embedding (part relation), shape difference, and addition. In computer implementations of shape grammars, embedding corresponds to a shape recognition problem. It defines the step where a certain part of an existing shape is identified to be then replaced by another shape. Shape difference is the removal of the identified part whereas shape addition is the summation of the leftover part and the new shape from the right side of the rule. Visually, for both rectilinear and curved shapes, these three steps are straightforward (Fig. 1).

In the implementations, however, shape embedding poses problems for freeform curves. Shape grammars are originally defined for a broad range of shape types (Stiny, 2006) but their computational implementations have mostly been for rectilinear shapes. Chau (2002), McCormack and Cagan (2003), and Jowers and Earl (2009, 2011, 2015) explored implementing shape grammars for curved shapes. Most of these studies adopt the maximal lines method Krishnamurti (1992) developed for rectilinear shapes. For the computational implementation of curved shapes, McCormack and Cagan (2003, 2006) use distinct shapes that are straight line equivalents of curved shapes. Jowers et al. (2004) discuss the detection of segments within a curve using maximal curve segments and their carrier curves where a carrier is an infinite curve within which another curve is embedded. In the case that a curve carrier is to be described piecewise, they show that a curve segment does not necessarily have the unique description normally required for computer implementations. Their suggestion for future work is the use of a combination of mathematical functions and curve properties to describe the carrier curves and curve segments. We propose a new implementation approach that allows for working with NURB curves without uniquely describing them, and to also freely select and transform any part without being limited to maximal curve segments. This method is intuitive and compatible with the way designers work with NURB curves.

Shape grammars value seeing sub-shapes when looking at shapes, including those that consist of free-form curves. Jowers et al. (2010) and Keles et al. (2012) have both relied on visual approaches in sub-shape detection. The latter approaches part relation entirely visually and, using a weight function, overcomes the limitations of working with canonically describable curves. This technique facilitates sub-shape recognition in shapes of any nature including free-form curves but is not yet implemented as part of common design software.

Whereas the main challenge in a shape grammar implementation for curves is in handling the part relation, it is just as important to recognize the challenges that working with curves

CAMBRIDGE UNIVERSITY PRESS

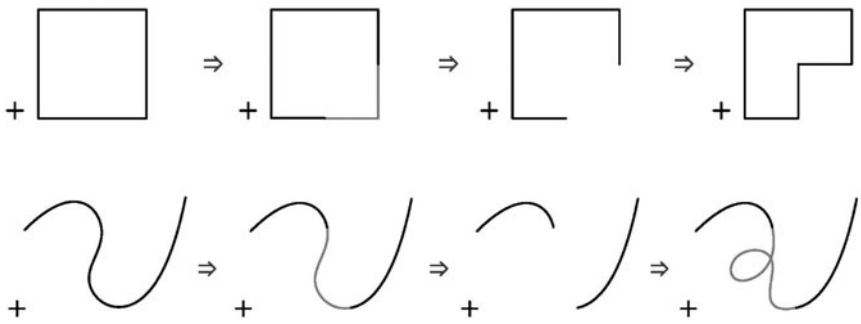

Fig. 1. Embedding, difference, and addition of rectilinear shapes and curvilinear shapes.

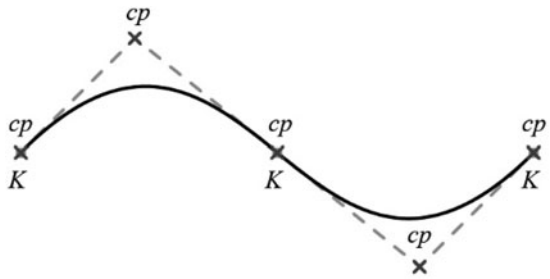

Fig. 2. A NURB curve. Control points, control polygon, and knots.

introduces to the designer. Jowers and Earl (2015) already note the vagueness in considering the embedding relation whether as analytical or as visual. Shape rule application in curved forms is not as straightforward as in the case of rectilinear forms. The eye is tolerant and allows for a broader class of similarities in curves. While adding curves, visual tolerance preserves the visual continuity of the whole(s) but not necessarily the character of the distinct curves. Visual continuity in the context of this paper is that there are no apparent breaks in a line that curves and any distinct points from which the eye is inclined to split that curving line into parts. The whole can be perceived as one maximal shape. In our approach, we focus on visual continuity where the added shape adapts to the existing shape so that the ends that touch are tangentially continuous with it. In tangential continuity, two curves share the same tangent at the touching ends.

Designers who work in digital modeling environments usually and repetitively manipulate a NURB curve through the displacement of its control points (from here on CPs). A curved shape can be formed and re-formed this way as naturally as one shapes an elastic material by hand. Continuous curved surfaces have become a part of the architect's vocabulary and facilitate a formal expression of flow between spaces, or response to contextual dynamics. The challenge posed by their construction has been taken on eagerly as a display of competency. A minimum of second degree (curvature) continuity is necessary for the manufacturing of smooth surfaces such as those in bodies of cars. Within the scope of this paper, we focus on two-dimensional (2D) curves. We look at the points where curve segments coincide and ensure first degree (tangential) continuity to preserve the visual continuity described above.

In order to ensure tangential continuity at touching ends of two or more curves in a shape rule application, we developed a tool in Grasshopper. Grasshopper is a plug-in for the NURB curve modeler Rhinoceros 3D (from here on Rhino) developed by McNeel, a commercial software commonly used by designers. We also utilize Python programming within the Grasshopper environment. Our tool follows three steps. First, the user manually selects a part of a NURB curve using two parameters. Secondly, that part is subtracted and thirdly, a new curved shape is added, which may be any curve separately constructed by the user. The newly added curve is unified into the initial shape preserving visual continuity. Our tool automates the performance of subtraction and addition with a visually continuous result.

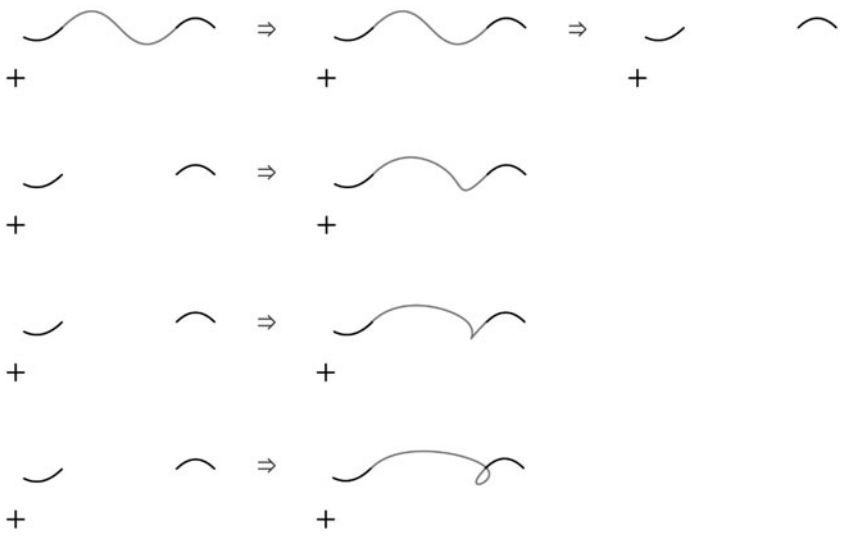

Fig. 3. Shape difference and additions of varying shapes.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

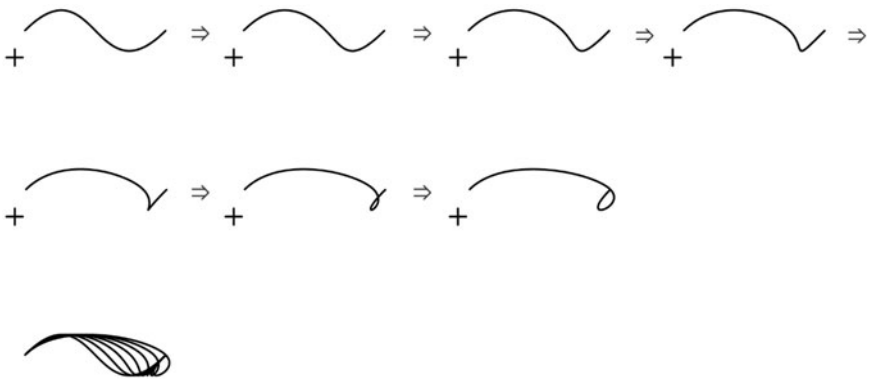

Fig. 4. Curve shapes obtained by moving one CP in the *x* direction, one unit at a time.

For shape subtraction, we rely on the splitting procedure available in the Rhino software environment. This procedure automatically adds CPs to the ends of the curve to preserve the shape of the remaining curves, useful in our tool for preserving the visual (tangential) continuity of curves that are worked with. This requirement, although visually consistent and functional, changes the mathematical construction of the NURB curve, by adding and rearranging its CPs. Our approach can be regarded as a practical application based on the visual processing of shapes, rather than an analytical one concerned with their formal structures. Nevertheless, some mathematical formulae are presented in the next section to explain how NURB curves are described.

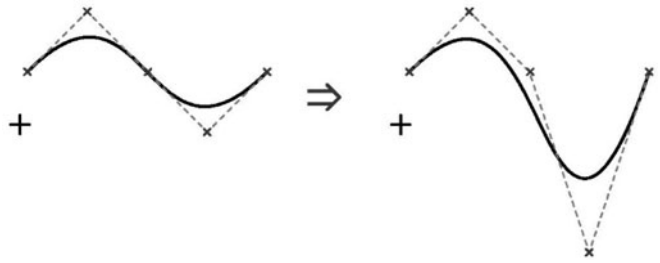

Fig. 5. Transforming one CP without disturbing the curve continuity.

Basic principles to facilitate shape grammar implementations of NURBs

NURBs are mathematical representations that can accurately describe any 3D geometry that may range from rectilinear planar shapes to complex freeform surfaces. This study focuses on 2D, or planar NURB curves, but the method proposed may also be expanded to work with 3D NURB curves and surfaces.

NURBs were invented by both Bezier and Casteljau around the same period, based on the traditional ship construction techniques that utilized controlled bending of bamboo strips called splines. The invention of NURB curves allowed for an effective representation of curved lines and surfaces. This proved to be very useful in

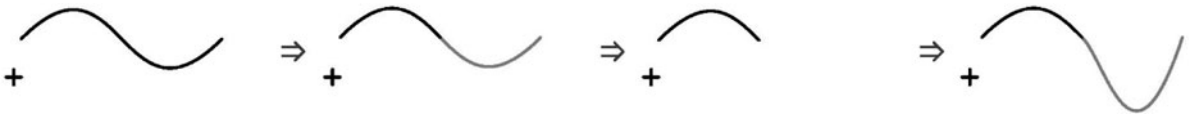

Fig. 6. Subtraction and addition of curve parts.

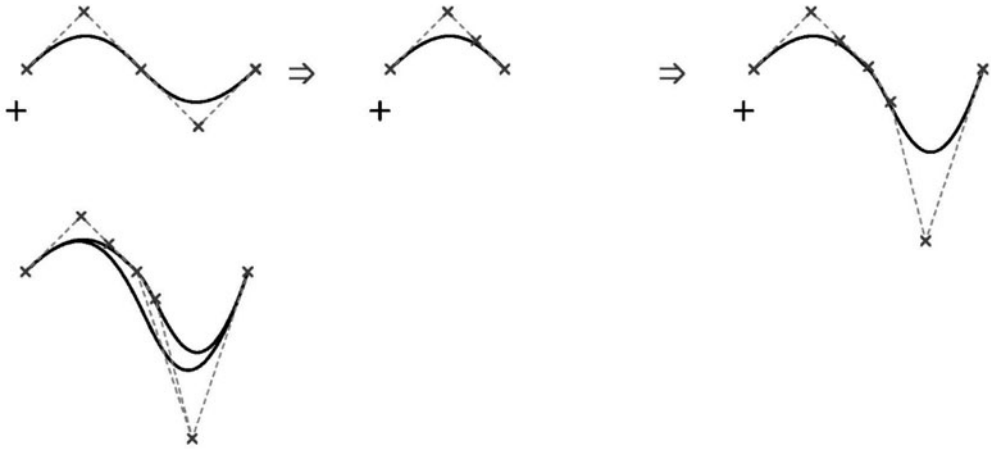

Fig. 7. The continuity of the curve is disturbed.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

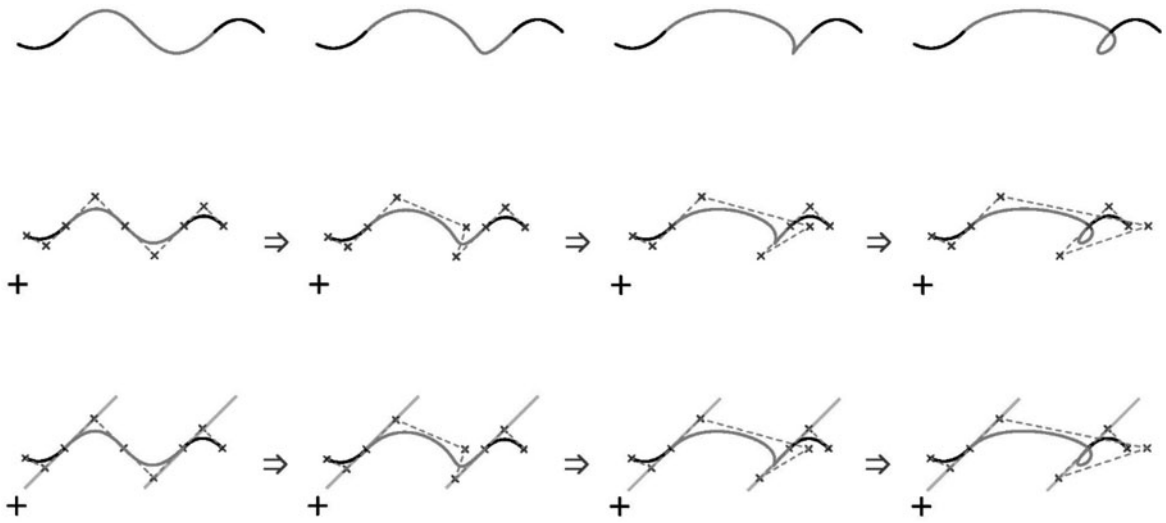

Fig. 8. Shape part is replaced preserving the continuity; CPs are activated and tangents are drawn at endpoints.

the design and manufacturing of cars, ships, and airplanes, which had aerodynamic forms. Quickly adopted by the design industry and incorporated into computer-aided design software commonly used by industrial designers and architects, NURB curve processors greatly extended the formal vocabulary of computer-aided design.

"NURB" is an acronym for "Non-Uniform Rational Basis-Spline." "Basis Spline" refers to a curve controlled by CPs (cp in Fig. 2) each associated with a basis function. "Rational" implies that the CPs can (but do not necessarily have to) have weights that are not equal to one and that determine the magnitude of the impact of each CP on the curve. The straight lines joining the CPs are referred to as the control polygon. "Non-uniform" stands for the free distribution of the knots (K in Fig. 2), in other words, the intervals determining the limits of influence of CPs along the length of a NURB curve. Knots are the joining points of the piecewise described curves making up the NURB curve. The following is the general mathematical equation for NURB curves (Piegl & Tiller, 1997).

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\, w_i\, \boldsymbol{P_i}}{\sum_{i=0}^{n} N_{i,p}(u)\, w_i}, \qquad a < u < b.$$

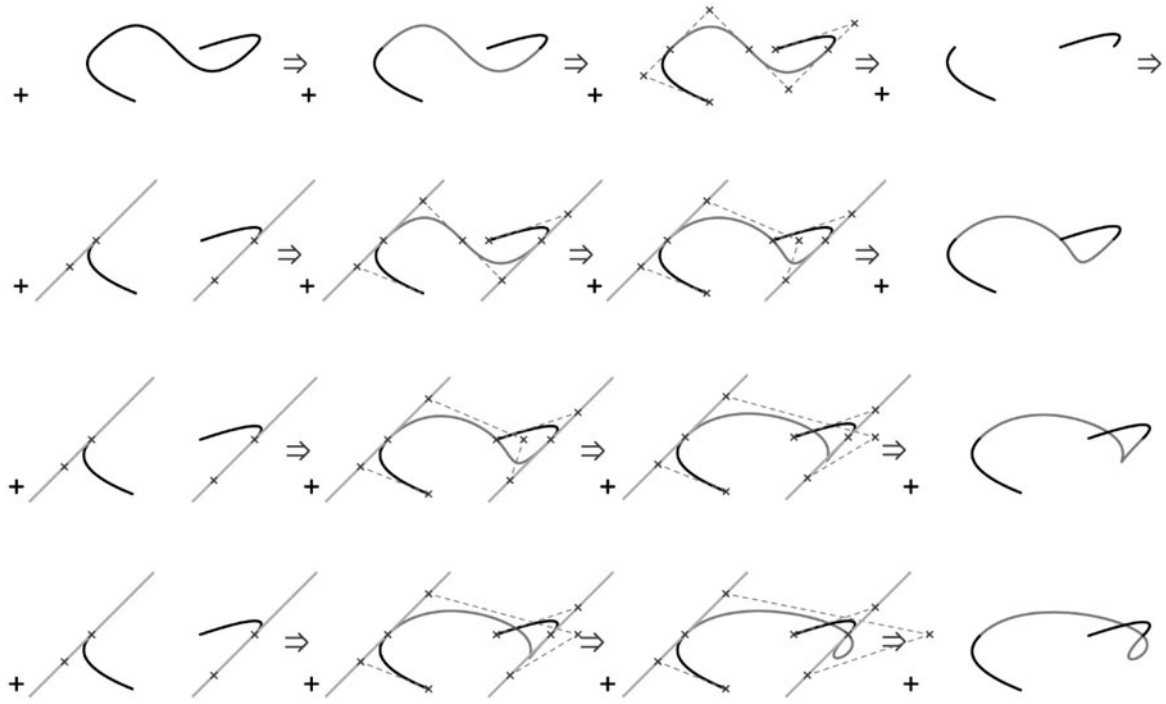

Fig. 9. Subtraction and addition of curve shapes preserving continuity.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

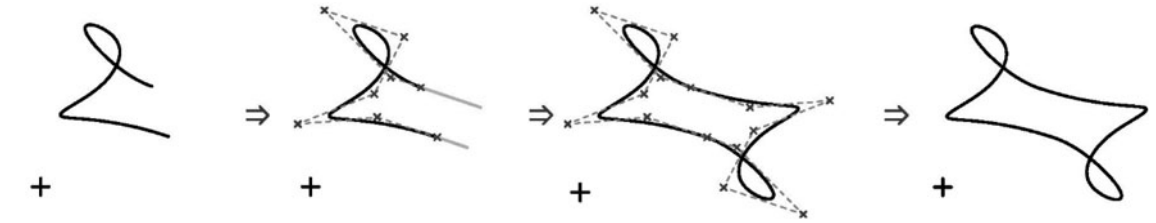

Fig. 10. Shape addition to create a closed curve.

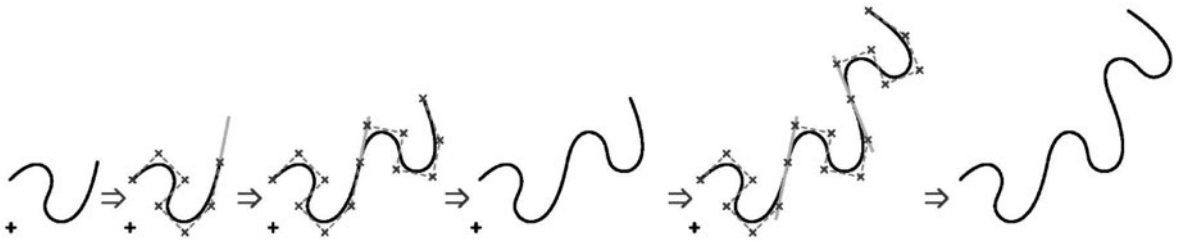

Fig. 11. Recursive addition with rotation at the ends of curves.

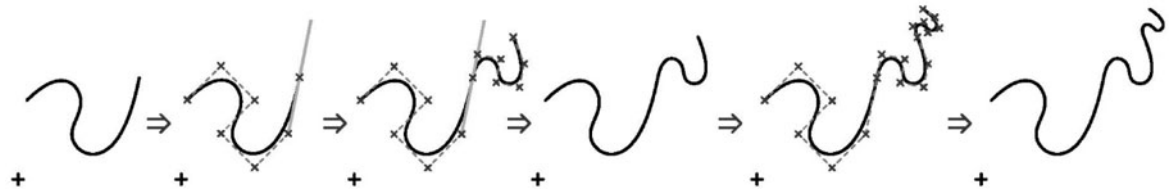

Fig. 12. Recursive addition with rotation and scaling at the ends of curves.

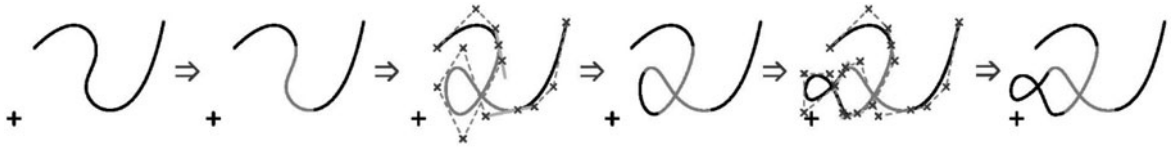

Fig. 13. Recursive subtraction and addition of curves within certain ranges of curves.

where $\{\boldsymbol{P}_i\}$ are the CPs, $\{w_i\}$ are the weights, $\{N_{i,p}\}$ are the rational basis functions of pth degree and the p indicates the degree of the NURB curve. In the case that u ranges from 0 to 1, or $a = 0$ and $b = 1$, assuming $w > 0$, we get:

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u)\, \boldsymbol{P_i}$$

where $\{R_{i,p}(u)\}$ are the piecewise rational basis functions on $u \in [0,1]$ (Piegl & Tiller, 1997).

The shape of the curve is determined by a polynomial interpolation over each CP. NURB curves are defined by an order, CPs, knots and an evaluation rule. The degree refers to the highest power of the polynomial function describing the curve and the order of a curve is always one higher than its degree. The evaluation rule refers to the parametric equation that gives the coordinate values of a point parameter on a curve.

The weights of all CPs of the curves used in this study are equal to one, which means the magnitude of influence for each CP is equal and they are *B*-spline curves. A first degree (tangential) continuity is preserved in the following examples of computer applications of shape rules with NURB curves in a visual programming environment.

Shape addition and difference in place of CP manipulation

In the Rhino modeling environment, to apply a shape rule to curves, the user subtracts (the left side shape) from an initial curve by splitting the initial curve. Splitting is a procedure defined in Rhino that requires the user to designate points of reference for where the curve will be cut. If the left side shape is to be subtracted from the middle part of the initial curve, the user will split the initial curve at two points. These are then the endpoints of the leftover shape and a gap remains in between them. Rhino automatically inserts additional CPs close to these ends of the

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

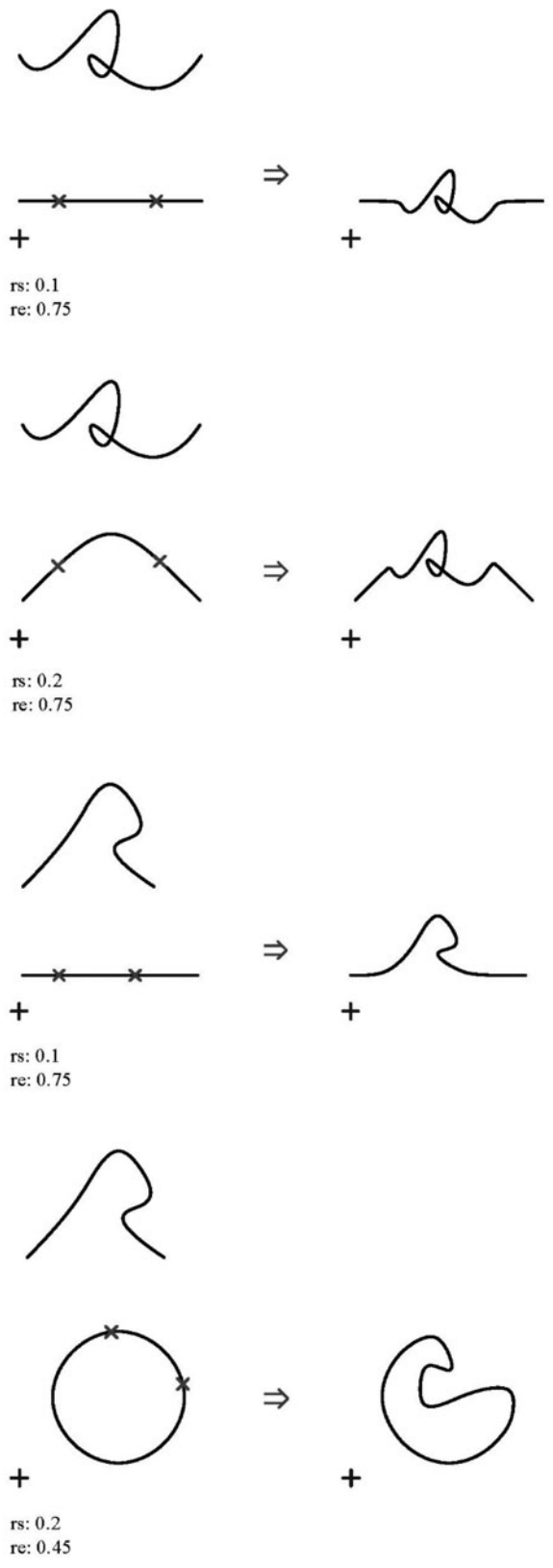

Fig. 14. The shapes at the top left side of each group are added within the defined ranges of the curves at the bottom left side to attain the shapes on the right side.

remaining curve to preserve its shape. Then, the curve that is going to be added should be scaled and rotated to fit the gap on the initial curve. Once it is positioned to fit the gap on the initial curve precisely, the blending command is used at both ends. Rhino provides options for this command to preserve the continuity of different levels and inserts new CPs at user-specified distances to the curve ends. Our tool automates this process using the Grasshopper plugin, allowing the user to subtract and add curves without having to repeat all these steps, substantially simplifying and speeding up an iterative workflow. It first requires that the user selects a range, delineating a section of the curve for subtraction and that he provides a second curve to be added to the initial curve. Our tool splits the initial curve at endpoints of the user-defined range with the splitting command in Rhino. It automatically scales and aligns the curve to be added to precisely fit the gap on the initial curve, then inserts new CPs at distances from the two ends to fit user-defined tolerances and finally re-draws the curve using the CPs of the remaining and newly added curve segments. When adding new CPs at joining ends, the tool positions them in order to preserve tangential continuity. As the added shape adapts to its context, also allowing variations to multiple instances, its context is preserved. We propose that such a tool can contribute to shaping computations in design processes that incorporate NURB curves.

A number of examples are presented here to demonstrate both the manual procedure in Rhino and our automated tool. We also illustrate how the manipulation of CPs is utilized in a regular workflow in Rhino which we propose to replace with shape difference and summation through the use of our tool. Figure 3 demonstrates the operation of shape difference, followed by three different additions of NURB curve shapes to generate new shapes.

The three new parts that are added to the shape at each step are the three CP-manipulated versions of the initial part that was subtracted from the original shape. As demonstrated in Figure 4, the three new shapes can, in fact, be generated, along with many more, by moving one CP of the initial curve, one unit in the *x*-direction at each step, without changing the positions of the rest of the CPs. Achieving the same results by shape subtraction and addition as intuitively and practically is the challenge we take on and facilitate. The juxtaposition of all versions of the shape is presented at the bottom of the figure.

Looking at the procedure of CP-manipulation more closely, Figure 5 demonstrates how a CP is moved to change the shape of a curve segment, without disturbing its continuity which would be the case if we were to subtract the second half of the curve and replace it with its manipulated version as demonstrated in Figure 6.

In Figure 6, first, a part of a shape is delineated, then it is subtracted from the initial shape, and a new curve is added in place of the subtracted curve.

Figure 7 shows that the new curve added has CPs coinciding with the CPs of the curve segment we have initially altered and still, the continuity has been disturbed. This is due to the addition of new CPs. Rhino has automatically added one CP to the existing curve segment to preserve its shape when being split, and one CP to the new curve which also happens to be a part of the altered version of the curve in Figure 5. These CPs allow for preserving the shapes of the parts of curves when they are split. However, subtraction and addition of curves bring about different results from manipulating their CPs.

There is a way to add and subtract curves seamlessly so that their continuity is preserved the same way when they are altered through the manipulation of their CPs. Below is the first example where the many altered versions of one curve segment replace a part of our original curve, preserving the shapes of the remaining

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

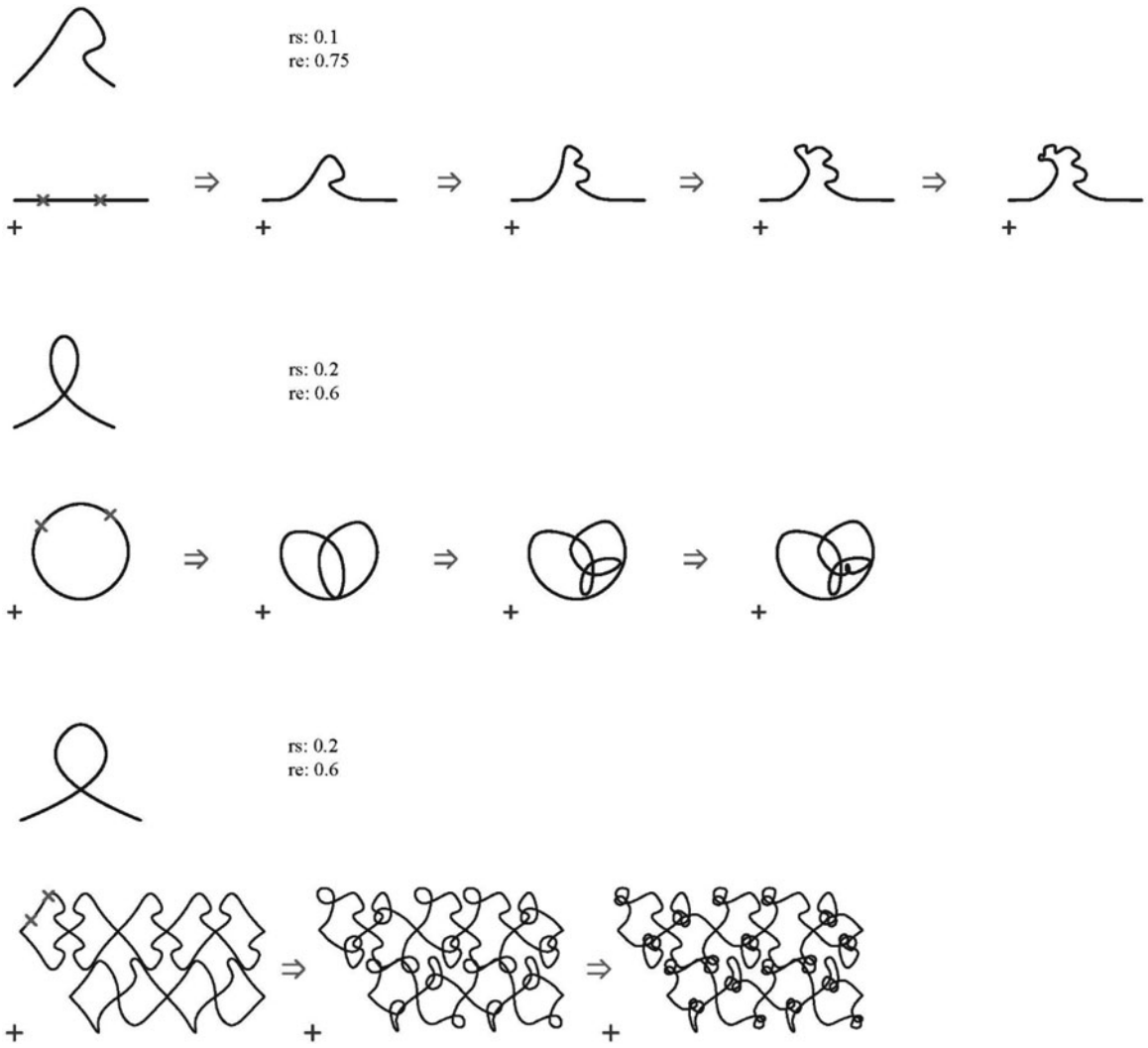

Fig. 15. The shapes at the top left side of each group are added within the defined ranges of the curves at the bottom left side and the initial range is repetitively remapped onto the new curve.

parts illustrated in black and the first-degree continuity in all of the resultant curves (Fig. 8 top row). Tangents of the curves at joining ends are equivalent.

To explain how this works, let us first look at how the CPs behave in the procedures performed (Fig. 8 second row). To add and subtract curves preserving their continuity is possible through the following set of conventions. As long as the tangent line at the end of the curve segment to be joined with a new curve aligns with the first two CPs of the joining end of the second curve, a new curve can be added to an end of any curve segment preserving both its shape and continuity. The CPs at the joining ends should coincide, and the second CPs only need to be aligned with the tangent line. All pairs of CPs and the joining ends remain on the tangent line at the joining end (Fig. 8 bottom row).

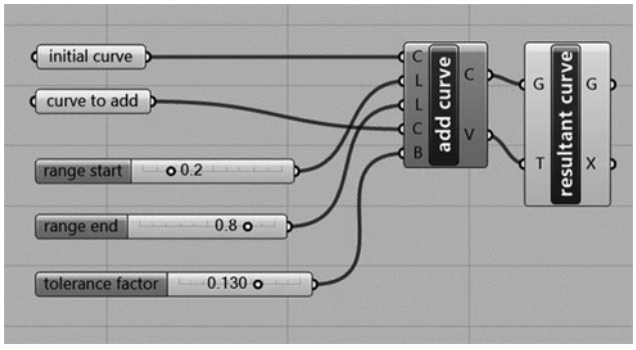

Fig. 16. Grasshopper definition allows for selection of range and tolerance value for smoothness.

To be able to subtract and add curve segments while preserving the continuity at the joining ends, we need to follow the steps below.

Step one: The segment to be replaced is delineated. Splitting procedure is carried out. CPs are activated, and the curve segment is deleted.

Step two: Using the two consecutive CPs at both ends of the curve segments on which the joining procedure will be performed, tangent lines are drawn.

Step three: A new curve segment that starts with two CPs coinciding with the tangent lines at the end of the first existing curve

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

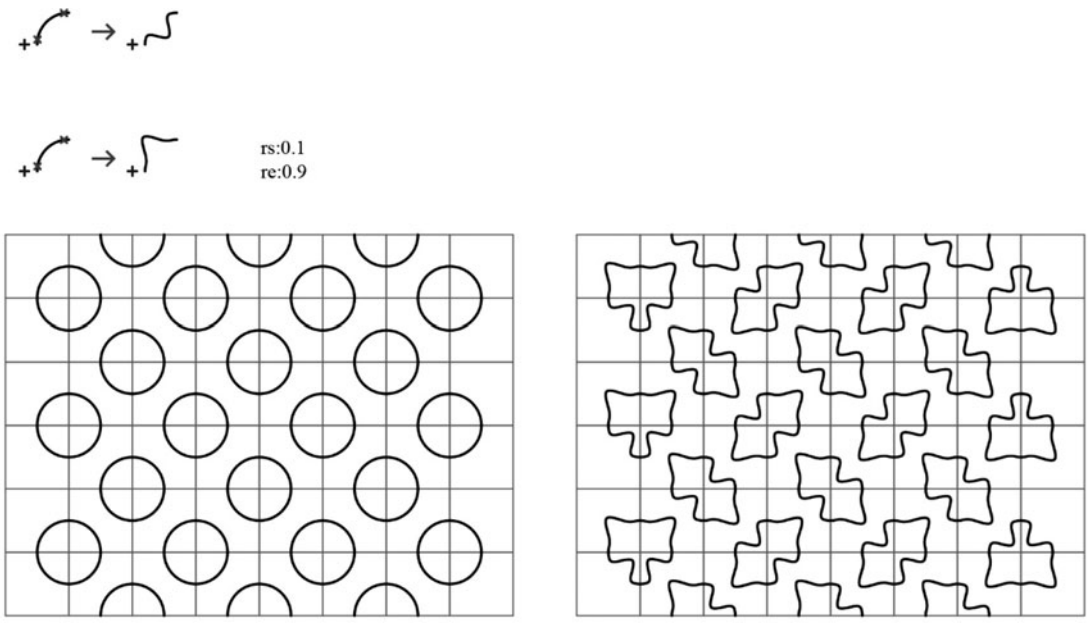

Fig. 17. Two different shape rules are applied to quarter circles.

segment, and ends with two CPs coinciding with the joining end of the second curve segment is added to the shape.

Step three can be repeated by the addition of many different curve segments (Fig. 9).

While the correspondence of the couples of CPs at the joining ends of curves is necessary, the matching of the number of CPs in between the ends of the replaced NURB segment and the new one is not a requirement to preserve the continuity of the whole curve. It is worthwhile to note here that the operations employed in our examples are not within a closed algebra since the new curve parts that replace the subtracted segments are new to our initial curve vocabulary. Jowers and Earl (2015) have introduced examples of closed algebras for curves. Notice that in step one, where we delineate the segment we will subtract from the initial shape, we apply a splitting procedure in the software environment. Visually, this would simply be embedding a part within the context of shape grammars. While visually this step does nothing to change the shape of the initial curve, it actually changes the mathematical construction of the NURB curve, by automatically increasing and rearranging its CPs.

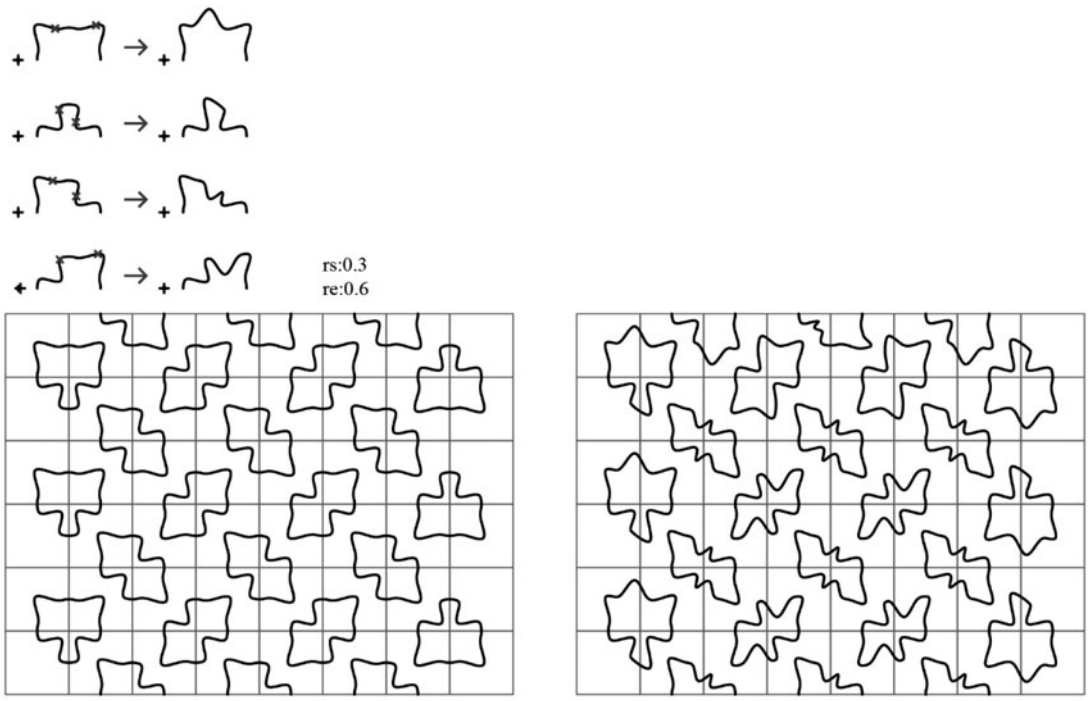

Fig. 18. Four different shape rules are applied to the parts of resultant curves of the previous example.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

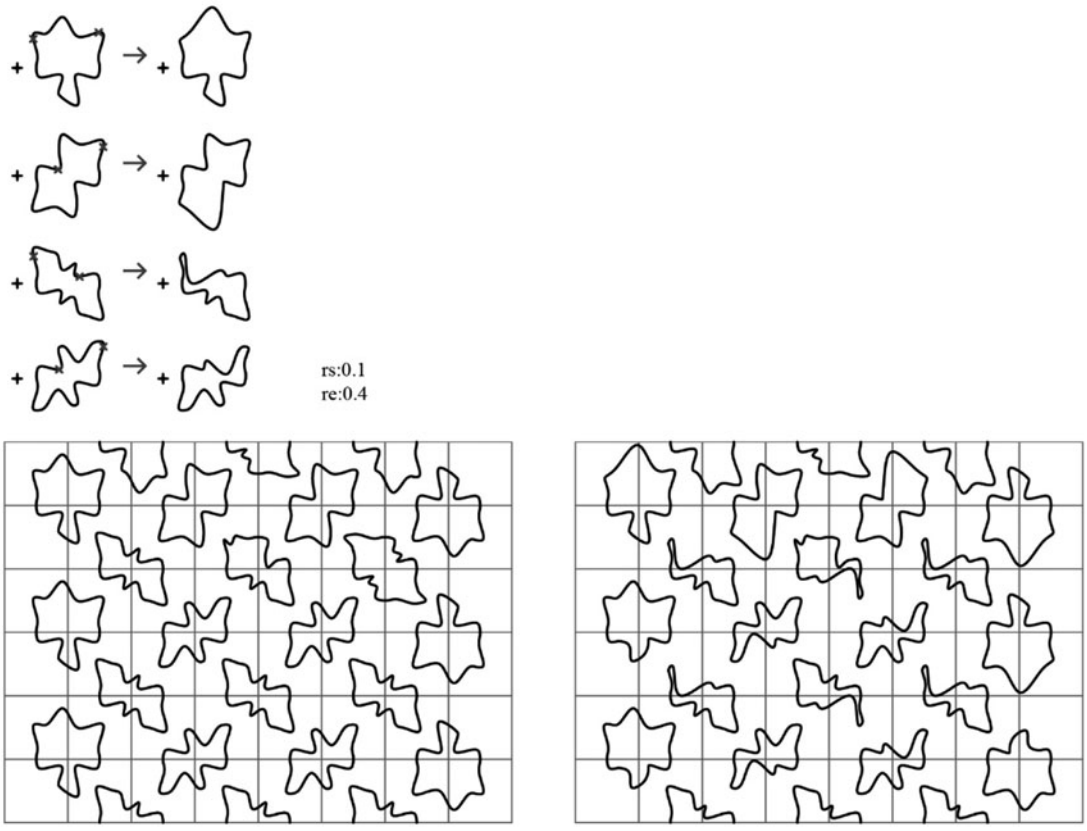

Fig. 19. Four new shape rules are applied to the resultant curves of the previous example.

The splitting procedure, performed at the boundary points of the part that is to be replaced, is necessary for the rules that follow. Splitting is necessarily a procedure performed through a command, algorithms of which are part of the NURB curve processor Rhino.

Following the rules defined above, both ends of a curve segment can be joined with other NURB curve segments, in the following example, constructing a closed curve. Again, the two CPs of the new curve align with the tangent lines at both ends. The applicability of the method to closed curve segments is meaningful considering that the 2D representation such as section cut drawings of 3D objects to be manufactured are always closed curved shapes (Fig. 10).

This procedure requires that if a new curve is to be added to the end of another curve, the tangent lines at both ends of the curve work as a guide to help satisfy the tangent alignment rule.

Application of the method with Euclidean transformations

Let us assume starting with a NURB curve segment as in the first shape in Figure 11. We can duplicate it, rotate and add this new curve segment to the end of the first curve, so that the two CPs at the joining ends align with the tangent of the first curve's joining end. This procedure can be repeated the same way Euclidean transformations can be applied to rectilinear shapes recursively through the original shape grammar rules (Fig. 11).

For another example, we can start with a NURB curve segment, duplicate it, transform its scale by 0.5 and rotate and add it to the end of the first curve, so that the two CPs at the joining ends align with the tangent of the first curve's joining end. Again, this procedure can be applied recursively, to get a fractal NURB curve (Fig. 12). Clearly, the requirement for the alignment of end CP tangents poses a limitation here: the rotation angles of the duplicated curves to be added at the ends are predefined to satisfy the rule and preserve the continuity of the curve.

The subtraction and addition operations can also be repeated following the rule to keep the two-end CPs aligned, to recursively add new curve segments within an existing NURB curve segment (Fig. 13). The examples presented in this section demonstrate the applicability of the procedure to any part of any NURB curve, allowing for the addition, subtraction and Euclidean transformations. This facilitates the use of shape grammars as a method when working with NURB curves.

In all examples presented in this section, the shape grammar formalism

$$S \rightarrow S - a + (b)$$

can be utilized to describe the steps followed. However, the shape to be added (b) is carefully selected or generated to satisfy the defined conventions that facilitate visual continuity of the resultant curves. At both joining ends of the curves, the two CPs of the new curve and the existing curve parts are congruent.

Implementation of the procedures in a visual programming environment

As results of a testing of the shape delineation, difference and addition steps of shape rule applications on NURB curves, this section illustrates computer-generated examples where we

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

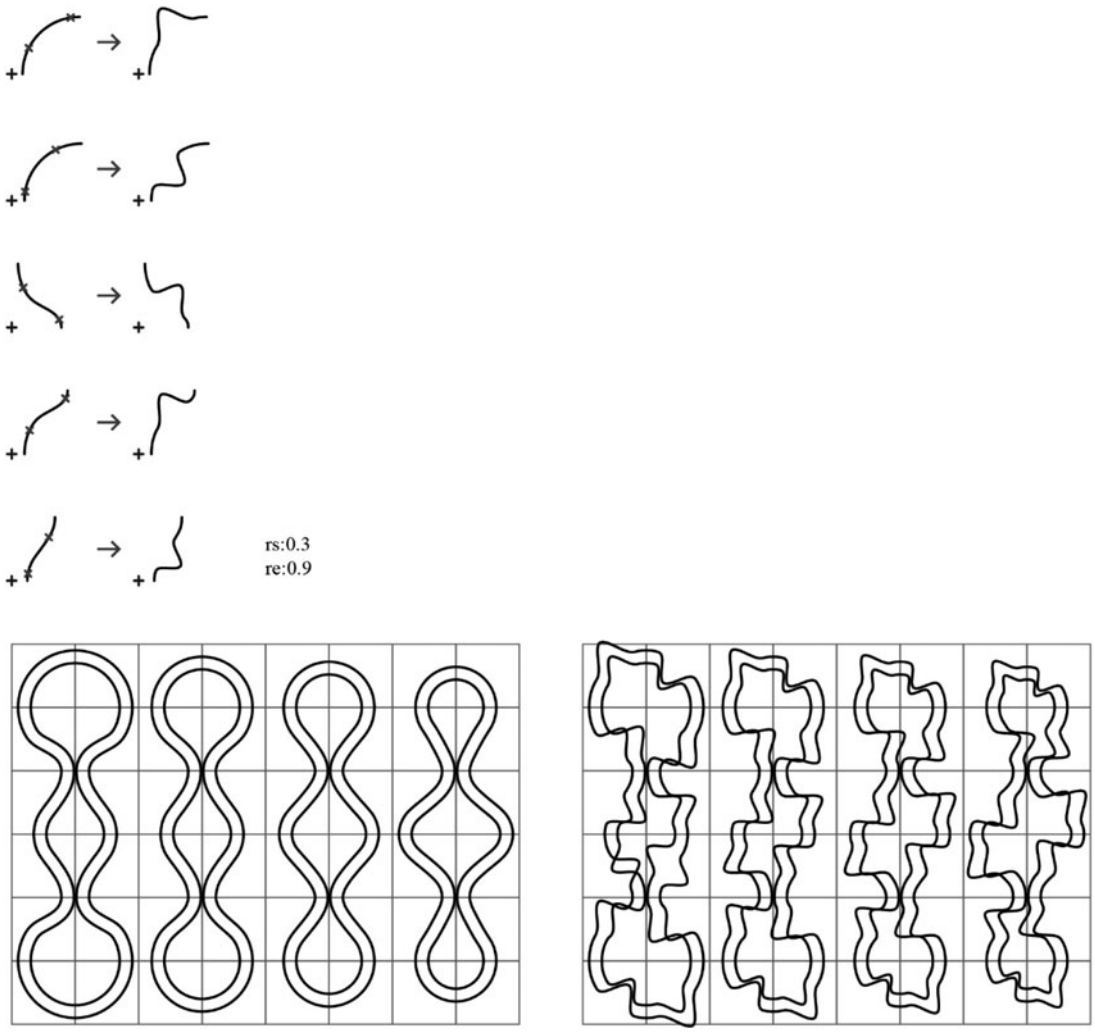

Fig. 20. Five shape rules are applied to initial shapes with parametric variations.

automate the above-explained procedures through our tool using the Grasshopper plugin.

The following is a list of the automated procedures in the examples to follow.

(a) Definition of a range on a curve, for the delineation of a part for shape difference. The user defines two values between zero and one. This step replaces the embedding step in traditional shape rule applications;
then
(b) Rotation, scaling, and repositioning of the new curve to be added to the initial shape;
and
(c) Redrawing of the curve with the insertion of additional CPs in between the curves, to preserve continuity.

The examples in Figure 14 demonstrate the shape difference and addition operations automated through the steps (a)–(c) defined above. "*Rs*" and "*re*" stand for the starting and ending points of the range defined by the user through our tool. This range can be between zero and one, zero denoted to the beginning, and one to the ending of the curve.

When the subtraction and addition operations are recursively applied, the range to subtract and add new shapes within the curve needs to be redefined. In the examples in Figure 15, the initial range defined was repetitively remapped onto the new curve, so that a fractal-like repetitive scaling and positioning was obtained. The last example demonstrates the application of the same procedure on a number of curves at the same time.

The interface in the Grasshopper environment allows for real-time manipulation of the range within the initial curve, automatically altering the scale and rotation of the new curves that are added to it (Fig. 16). The tolerance factor determines the distance of the CPs that are inserted to the starting and ending points of the range. The larger the tolerance defined by the user, the more continuous the curve is perceived. This is due to the increased distance of the CPs from the joining ends in the tangent direction at these points.

What deviates from the analog shape grammars approach in these examples is that some curve segments are inserted between the remaining shape and the new curve that is added, to blend the curve segments into a continuous whole. These segments are necessary for the preservation of visual (tangential) continuity of our curve.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

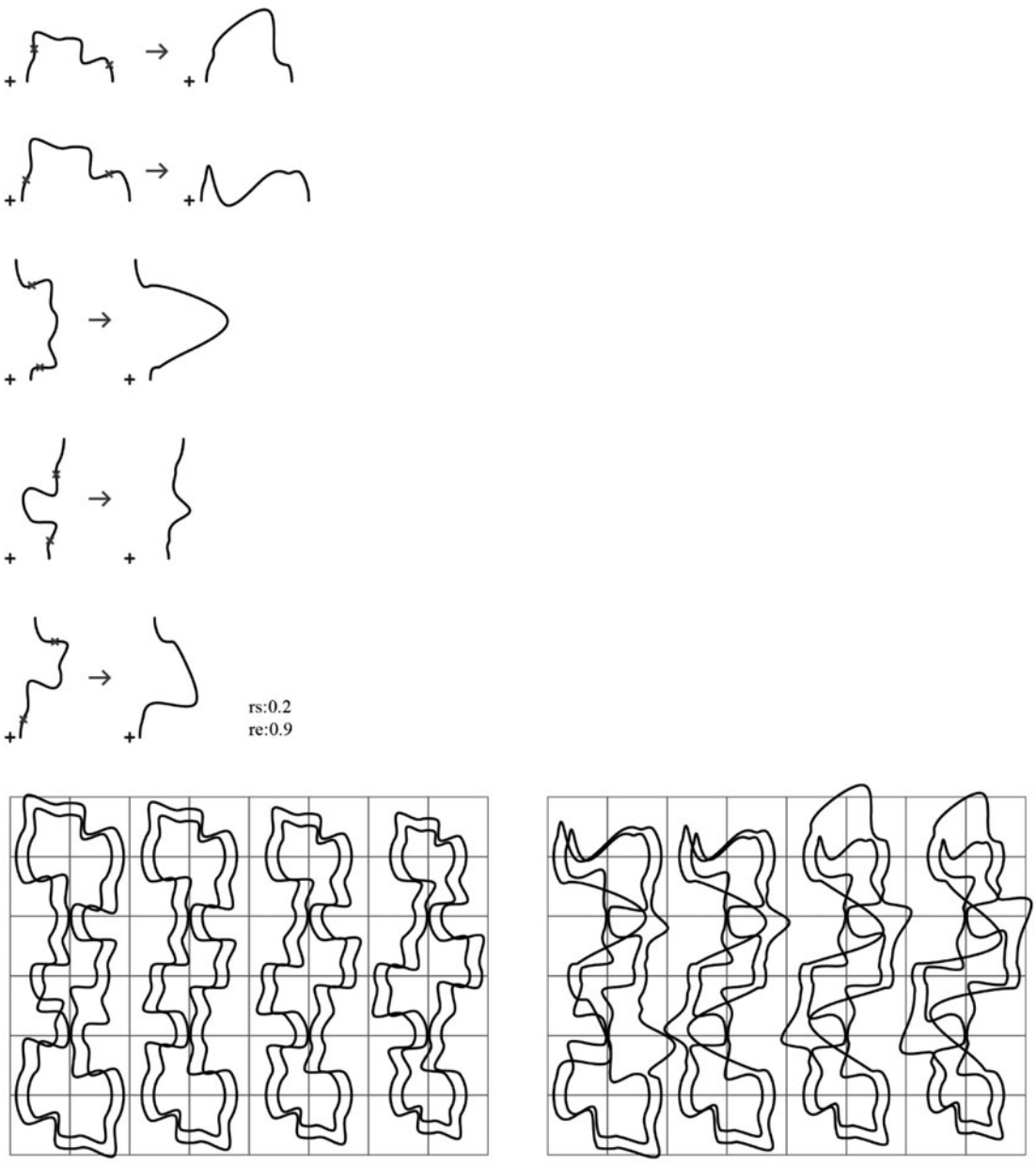

Fig. 21. Five new shape rules are applied to parts of the resultant curves of the previous example.

Application of the method on a design problem: seamless tile motifs

In this section, we present a set of examples where the Grasshopper tool we have developed to automate our workflow is used to design some seamless tile motifs. The idea is that a simple base of continuous curved shapes is arranged as a repeatable tile motif, and variations of new curved patterns are obtained preserving the continuity of each shape while the tile design remains repeatable. The choice of examples in the form of tile motifs provides an underlying grid canvas to easily demonstrate the variations within curved shapes in a uniform system.

Initial curved shapes are manipulated using the same set of steps previously defined in the section "Implementation of the procedures in a visual programming environment" of this text. Certain ranges within each curve segment are delineated as explained in the section "Shape addition and difference in place of CP manipulation". The parameters determining these ranges have been provided with the images. Then each segment is removed and then replaced by a new curve segment, similarly with the shape grammar subtraction and addition, however, as defined in the previous section, additional curve segments are generated to preserve visual continuity.

In Figure 17, two shape rules are applied to the initial curves which are quarters of circle arcs, adding in both cases, the same shape at the top in two different rotations. In Figure 18, the initial shapes are parts of the curves that were the resultant shapes of the previous example (Fig. 17). Here, there are four different shape rules, each applied to a different shape. In Figure 19, the resultant curves of the example in Figure 18 are taken as initial shapes, and again, four different shape rules are applied to them. In all these three examples, the rule is obtained by the subtraction of the delineated parts (the ranges of which are indicated at the bottom right

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

of shape rules) followed by the addition of the same curved shape at the top in Figure 17. This shape is added to the left side of the curves by our tool, as previously demonstrated in the examples in Figures 14 and 15. Notice in all these examples that while the shapes on the left side of the rule remain unchanged, the shapes on the right side do not consist merely of the remaining part and the added curve. Additional curve segments have been generated that blend the curve parts together, the lengths of which are determined by the tolerance factor, adjustable as a feature of our tool.

Figures 20 and 21 show further exploration of the same workflow, where multiple rules are generated by the subtraction and addition of the same shape at the top in Figure 17 into initial shapes making up seamless, repeatable patterns.

Figure 22 illustrates the use of the same workflow. However, this time the rules are generated with seven different curved shapes added to the curve segments following the subtraction of the delineated parts. Notice that not all parts of the initial shapes in the final example (Fig. 22) exactly match with the initial shapes in our shape rules; however, they are versions of the same curves with an identical number of CPs, altered through the manipulation of their CPs.

The examples above demonstrate that the continuity and repeatability of the final pattern depend on the initial shapes. Once these requirements are satisfied for the initial shapes, the resultant patterns generated by the application of the rules do not break these requirements.

Our automated workflow, actualized in the tool, can be helpful in working with NURB curve geometries, especially when visual continuity is a concern, and the designer is seeking to apply shape rules to explore variations and experiment with families of curved shapes. Starting with relatively simple curves that are easier to manipulate and then generating new shapes based on

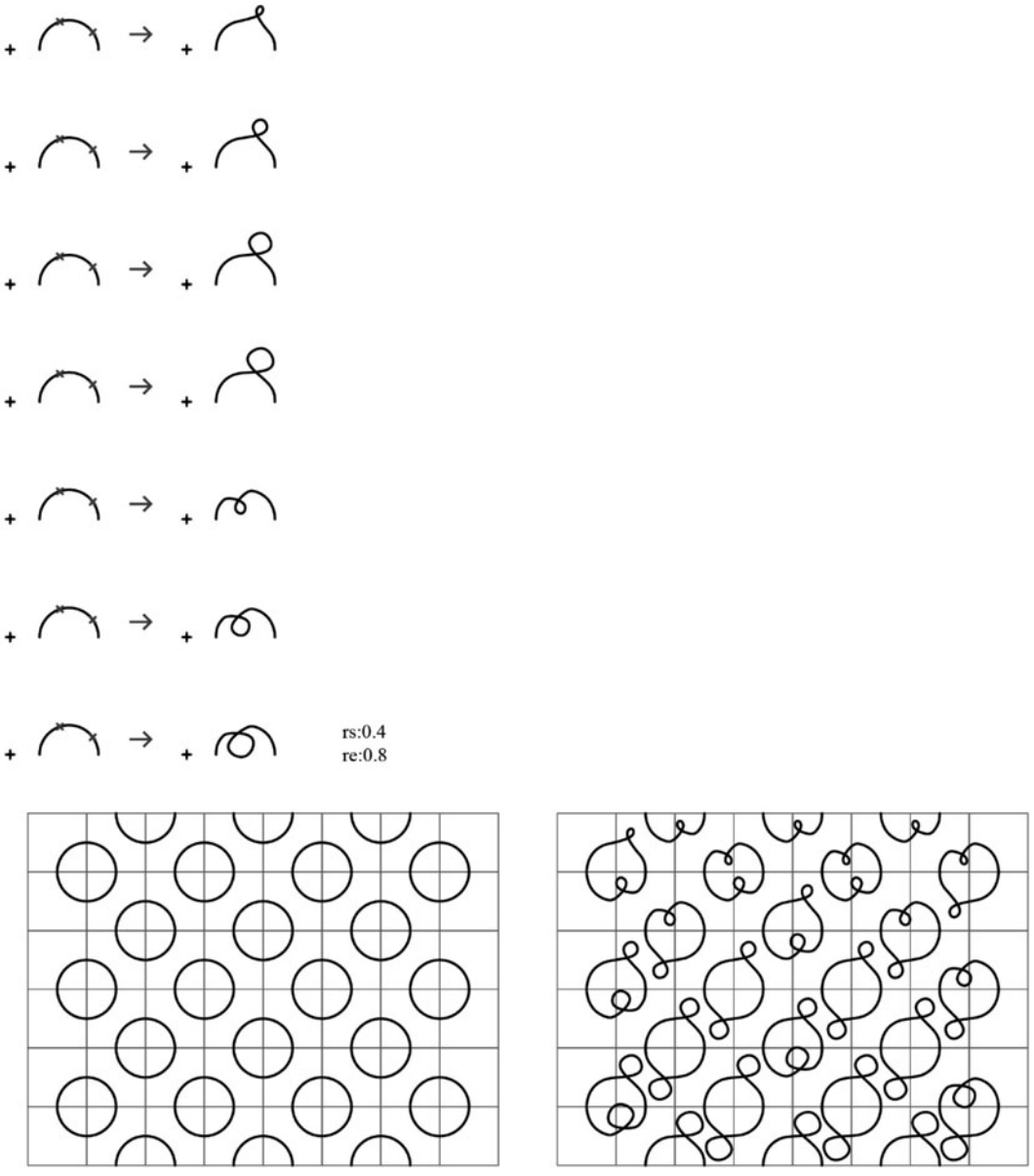

Fig. 22. Seven different rules are applied to half circles in rows.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press

these curves with the presented workflow will simplify and speed up the process of working with curved shapes, as well as providing rapid multiplication of variations.

Conclusion

The properties of NURB curves challenge computer implementations of shape rule applications. Nevertheless, we show that it is possible to incorporate these two notions in a partial solution. Instead of a complete implementation, the contribution here is a modest tool programmed within an existing design platform that enables automated workflows with NURB curves in shape computations as part of common design processes. Our approach automates steps in the common design modeling environment Rhino. These steps are namely splitting a NURB curve, scaling, rotating and aligning the new curve part, adding CPs and redrawing the resulting NURB curve. By way of this tool, any NURB curve segment regardless of type or mathematical description can be added to any curved shape, guaranteeing a continuous resultant curve. We have shown multiple rule applications and multiple instances of parametric rule applications. While the cases up till Figure 13 demonstrate our steps for NURB curve addition and principles behind them, the examples in Figures 14 and 15 show multiple versions of shapes that can be generated using our presented design tool. The examples in Figures 17–22 show how our tool can be used in a design problem to generate seamless tile motifs.

There are some limitations to the presented method. Firstly, we do not utilize an analytical approach for shape recognition but suggest alternative means to delineate curve parts by defining a range. As this requires numeric specification, it is a step away from a visual selection. Nonetheless, due to the nature of freeform curves, any part of a curve can be removed, and any new curve can replace that part, as long as a visual or, as specified in this paper, tangential continuity is preserved throughout the initial and resulting curved shapes. Secondly, while adding and rearranging the CPs of a NURB curve yields results that are visually consistent and functional, it actually changes the mathematical construction of the NURB curve. Nonetheless, the addition and removal of CPs without disturbing the ones neighboring the joining boundaries visually preserves the parts of the shapes that are not replaced and work for the purposes described in this paper. Also, the requirement to preserve the tangent lines at the boundary points poses a limitation in the Euclidean transformation of rotation as it allows only for a specific angle. In the case that a curve is added within a range of another curve as opposed to being added to its end, the Euclidean transformation of scaling is also limited by the distance between the two points defining the range within the curve.

The proposal in this study is only for planar NURB curves with equally weighted CPs but can be extended to 3D NURB curves and surfaces in future work. Further development of the method into a NURB curve implementer, followed by its testing on simple basic design problems in a workshop with experienced designers or an educational environment with novice designers may be fruitful in understanding its presumed practicality and applicability for shape computation in professional use and pedagogical agendas.

References

Chau HH (2002) *Preserving brand identity in engineering design using a grammatical approach (Doctoral dissertation)*. University of Leeds, Leeds, United Kingdom.

Jowers I and Earl C (2009) The construction of curved shapes. *Environment and Planning B: Planning and Design* **37**, 42–58.

Jowers I and Earl C (2011) Implementation of curved shape grammars. *Environment and Planning B: Planning and Design* **38**, 616–635.

Jowers I and Earl C (2015) Extending the algebras of design. *Nexus Network Journal* **17**(3).

Jowers I, Hogg DC, McKay A, Chau HH and de Pennington A (2010) Shape detection with vision: implementing shape grammars in conceptual design. *Research in Engineering Design* **21**, 235–247.

Jowers I, Prats M, Earl C and Garner S (2004) On curves and computation with shapes. In Akin O, Krishnamurti R and Lam KP (eds). *Generative CAD Systems Symposium: G-CADS 2004*. Pittsburgh: Carnegie Mellon University, pp. 439–457.

Keles HY, Özkar M and Tari S (2012) Weighted shapes for embedding perceived wholes. *Environment and Planning B: Planning and Design* **39**, 360–375.

Krishnamurti R (1992) The maximal representation of a shape. *Environment and Planning B: Planning and Design* **19**, 267–288.

McCormack JP and Cagan J (2003). Increasing the scope of implemented shape grammars: a shape grammar interpreter for curved shapes. In *Proc. ASME 2003 Int. Design Engineering Technical Conf. & Computers and Information in Engineering Conf.*, Paper No. DETC2003/DTM-48643, Chicago, IL, September 2–6.

McCormack JP and Cagan J (2006) Curve-based shape matching: supporting designers' hierarchies through parametric shape recognition of arbitrary geometry. *Environment and Planning B: Planning and Design* **33**, 523–540.

Piegl L and Tiller W (1997) *The NURBS Book*, 2nd edn. New York: Springer-Verleg.

Stiny G (2006) *Shape: Talking About Seeing and Doing*. Cambridge, Massachusetts: MIT Press.

Elif Ensari is a PhD candidate at Istanbul Technical University, in the Architectural Design Computing Program, and at University of Lisbon, Faculty of Architecture. Her current research interests include shape grammars implementations and computational tools for assessment and design generation for urban design. She holds a BArch degree from Middle East Technical University and an MArchII degree from Southern California Institute of Architecture. She is a part-time scholar at Istanbul Bilgi University and co-owns Iyiofis, an architectural design practice based in Istanbul.

Mine Özkar is the Coordinator of the Architectural Design Computing Program and a Professor of Architecture at Istanbul Technical University. Her research focuses on visual, spatial, and material aspects of design computation, and their integration to foundational design education. Her own teaching of computation theory and studios at both undergraduate and graduate levels is centered on students' critical thinking and involved doing. Mine completed SMArchS in design inquiry, PhD in design and computation at MIT where she was a visiting professor for a semester in 2013.

https://doi.org/10.1017/S0890060417000592 Published online by Cambridge University Press