Continuous Search & Replace in Vector Graphics

J. Loviscach

Fachbereich Ingenieurwissenschaften und Mathematik, Fachhochschule Bielefeld (University of Applied Sciences), Germany

Abstract

Many types of vector graphics comprise large numbers of almost identical partial shapes such as serifs in typeface design, ornaments in illustrations, and stylistic elements in icon design. As of today, later design changes of such shapes require the user to edit every instance on its own. Standard features of vector graphics software help with the automated replacement of stand-alone graphical objects, but do not cover partial shapes. Hence, a geometric search & replace function is needed, which is exacerbated by the circumstance that the repeated shapes' construction from curve segments tends to vary from instance to instance, which requires a continuous approach that does not rely on a fixed division of a path into segments. This paper presents a geometric search & replace function that addresses these issues, possesses adjustable tolerance against orientation and size changes between the search pattern shape and the retrieved shapes, and is integrated with a standard vector graphics program.

Categories and Subject Descriptors (according to ACM CCS): I.3.4 [Computer Graphics]: Graphics Utilities—Graphics Editors

1. Introduction

Most current vector graphics software allows reusing shapes in the spirit of a link in a file system: Editing one instance will change all of its copies, provided that a special cloning function—such as creating a "symbol" in Adobe Illustrator or a "clone" in Inkscape—has been applied to create the duplicates. Identical shapes generated without this function are not recognized. On top of that, *parts* of paths cannot be searched for and replaced. This is particularly vexing for ornaments that occur within other paths. This paper presents a method to address these issues. As repetitions may occur in orientations and at scales different from those of the original shape, the method is built to be invariant to such changes, see Figure 1. However, the user can limit by how many angular degrees the original shape may be rotated left or right and by which factor it may be shrunken or enlarged.

Graphical search & replace has been demonstrated before [KB88], but was limited to subpaths that span entire segments of a path. The method presented here also covers subpaths that start and end anywhere on a curve, hence the name "*continuous* graphical search & replace." This requires an entirely different approach, which is inevitable for many practical applications, as geometrically similar instances of a subshape may possess highly different internal structures in terms of (Bézier) segments, see Figure 2.

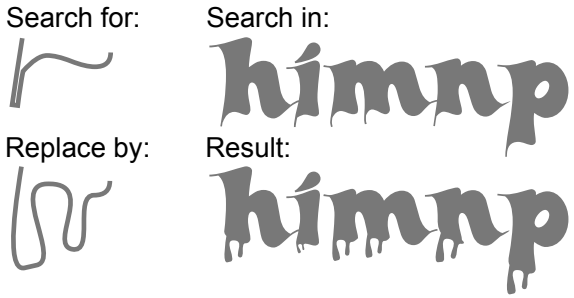

Figure 1: *Typefaces—extravagant or not—contain many ornaments to which search & replace is applicable.*

The overall approach is the following: The user inputs a collection of open or closed vector shapes in with the search & replace process has to be conducted, one open path that acts as search pattern, and one open path that the found subpaths are to be replaced with, appropriately sized and orientated. A feature extraction algorithm that identifies salient points is applied to the search pattern and all paths that are subject to the search. These salient points are applied to estimate the size and orientation in a partial matching test. A final test for matching is based on rendering and compar-

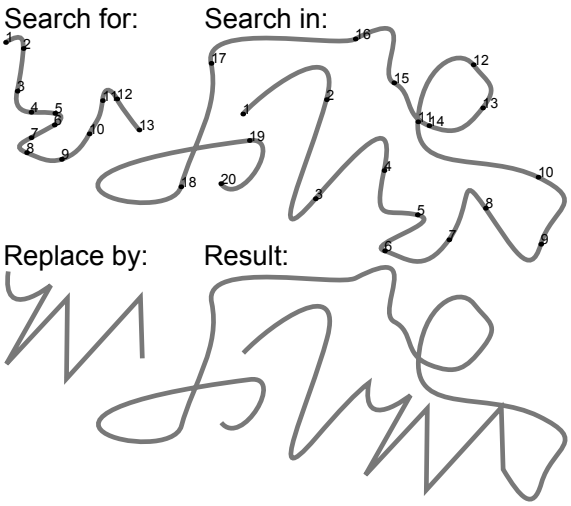

Figure 2: *The internal structure of the paths used for matching may differ highly; the geometry may differ slightly (numbered points indicate Bézier segments).*

ing fat curves. Up to here, the found matches range from one feature point to another; then, the matching regions are extended to arbitrary points along the paths. At this stage, the replacement takes place. Since a matching subpath may start and end anywhere on a path, the path is split appropriately before splicing in the replacement, which is rotated and scaled according to the original subpath.

This paper is structured as follows: Section 2 discusses related work, Section 3 describes the geometric features that are employed for the matching process, which is detailed in Section 4. Section 5 covers the replacement of the retrieved similar subpaths. The prototype is treated in Section 6. Section 7 concludes this paper, outlining future work.

2. Related work

Kurlander has addressed graphical search & replace in a number of works [KB88, KF92], treating an integer number of segments of a path. This allows straightforward methods to normalize the shapes in terms of position, size, and orientation. Seemingly, the topic has not been taken up any more after these works. In a work of similar age, Herz and Hersch [HH93] sketch how one could find repeated parts in typeface characters by an approach that resembles a vector quantization of partial shapes. In a follow-up work [HHGH98], they propose to assemble typeface characters from components such as serifs whose parameters have been determined by automated measurement. More recently, Hertzmann et al. [HOCS02] looked into "curve analogies," a method to built curves from examples through techniques that resemble those applied in texture synthesis. Including only translation and rotation, their method can employ different methods than proposed here.

Several methods have been introduced to identify salient points on curves, see e. g. Fischler and Wolf [FW94]. Mori et al. [MBM05] define "Shape Contexts" that store a fingerprint of the curve's local behavior in a histogram. Manay et al. [MHC*06] define integral invariants to regularize a notion related to curvature. Both works target shapes derived from bitmap images which typically are rough from noise. This is different from typical "designed" shapes found in vector graphics to be addressed here.

Shape matching has gained wide attention in 3D object retrieval [TV08], which employs methods that only superficially resemble those of 2D shape retrieval [VH01]. Pauly et al. [PMW*08] propose methods to find regular transformation patterns in 3D data; the problem presented in this paper lacks the repetitiveness of the transformations.

3. Feature extraction

Since trying to matching any point on one path to any other point on every other path is not viable, one has to reduce the matching problem to a finite number of tests. Following a general approach in shape matching, salient points are sought. The computation of these must be invariant under similarity transformations and reparameterization. In addition, it must be robust against small changes in shape.

Curves in hand-made vector graphics rarely contain strong noise, as this would require to manually create a huge number of curve segments. The method proposed in this paper caters for this application. It is known [FW94, MHC*06] that general curves are too rough to look for differential features such as inflection points or points with maximum curvature. However, in initial experiments also typical curves of vector graphics turned out to be not amenable to such methods. The number of inflection points is astonishing low, and using the points with maximum curvature was also ruled out, as these tended not to be robust, even with different kinds of smoothing including a Gaussian or a bilateral filter applied to the turning curve (that is, the mapping from arc length to tangent direction) and including constructing an approximate osculating circle from three points on the curve. One basic issue that standard vector graphics pose are discontinuities (often visually imperceptible) of the curvature between one Bézier segment and the next.

Given the problems with curvature, the proposed algorithm uses the turning curve directly. First, jumps are found that exceed 10°. The corresponding points on the curve are immediately considered salient. Next, the (continuous or almost continuous) evolution of the turning curve between these jumps is considered. This is approximated by a polyline that may only deviate up to 20° from the turning curve. This polyline is built from the coarse to the fine scale: It

is recursively subdivided at the location of the maximum error. This can be considered a piecewise approximation of the curve by circular arcs, as non-horizontal line segments in the turning curve correspond to circular arcs in the actual curve.

Not these points of subdivision are employed as salient points. Rather, the points midway between them (in terms of arc length) are used. As turned out in experiments, the points of subdivision lie a relatively horizontal parts of the turning curve, that is: in regions of low curvature. The regions of high curvature—that is: fast changes of the direction—fall between these points, see Figure 3. To exclude relatively flat regions, all parts of the polyline that do not cover an angle of at least 45° are discarded. For a collection of examples see Figure 4. Note that all computations handle the wrap-around needed for closed paths. In addition, types of curves other than Bézier could be handled with the same algorithm.

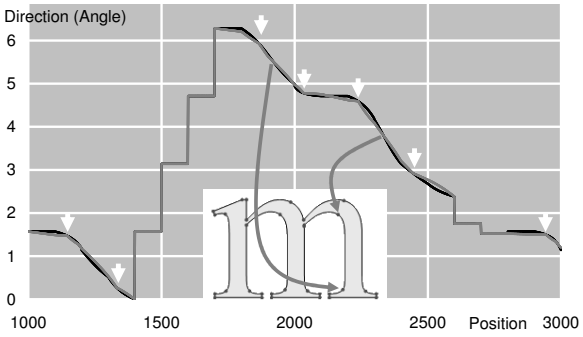

Figure 3: *The turning curve (black) is approximated by a polyline (gray, breaks indicated by white arrows). The midpoints of linear segments that cover more than 45° on the vertical axis serve as feature points. Note that the horizontal axis is given in 100 steps per Bézier segment, whereas the polyline is computed in terms of arc length.*

Figure 4: *The computed salient points reliably indicate high curvature with little algorithmic tweaking and only two uncritical parameters (examples from well-known typefaces).*

4. Matching

In principle, one could use dynamic programming to determine a error-tolerant correspondence between the feature points on the (sub-)path that is to be searched for and the path that is currently being examined for an occurrence, see e. g. [MHC*06]. Given the low number of feature points being considered and aiming at a high reliability, this paper proposes a different approach:

Starting from the ordered list of feature points $\mathbf{x}_i$ along a subpath, the algorithm computes vectors and numbers characterizing the position $\mathbf{p}$, size s, and orientation angle ϕ of the subpath traced by these points:

$$\mathbf{p} = \sum_i w_i \mathbf{x}_i, \quad s^2 = \sum_i w_i \mathbf{x}_i^2 - |\mathbf{p}|^2,$$

$$\phi = \mathtt{atan2}\left(\sum_i w_i h_i (\mathbf{x}_i - \mathbf{p}) \right),$$

where w_i is an appropriate weighting function with $\sum_i w_i = 1$, and h_i increases from -1 for the first feature point to 1 for the last one, growing linearly with the arc length covered. The motivation behind these quantities is as follows: $\mathbf{p}$ generalizes the centroid, s^2 the variance, and ϕ can be considered the direction of a generalized first moment.

To be robust against missing or added feature points, the weights w_i involve the arc length between the feature points: Every one of them is weighted by the half of the arc length between it and the previous feature point on the curve (if present) and half of the arc length to the subsequent one (if present). These weights are normalized by dividing by the total arc length between the first and the last feature point in the subpath that is considered. Based on the n feature points of the search pattern, this computation is executed in an exhaustive manner on all $m = n - 3$ through $n + 3$ consecutive feature points anywhere on a path in the collection to be searched through. The number n typically cannot be matched exactly as some feature points may be missing or be added due to small variations in the shapes.

The values for the size and orientation of the subpaths from the vector graphic are compared with the ones of the curve that is searched for. A subpath is rejected at this stage if the ratio of the sizes or the difference of the angles is outside of the range specified by the user. Otherwise, the Bézier curves forming the subpath are transformed to match the position, scale, and orientation of the path that is searched for. To determine whether the curves actually coincide, the transformed path is rendered as a fat curve into a bitmap of 200×200 pixels. The algorithm then checks if the path that is searched for is covered by this fat curve, see Fig. 5. The pixel positions that make up the path that is searched for are collected once, on startup. In the end, a list of validated instances of the search pattern is produced, together with the necessary similarity transformations.

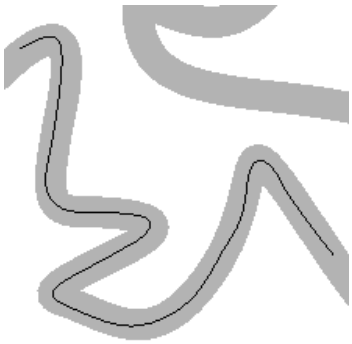

Figure 5: *A possible match is validated by drawing the transformed path as a fat curve (gray). This has to contain all pixels of the path forming the search pattern (black).*

5. Replacement

The replacement step of the algorithm works on the list provided by the matching step. First, it cleans up overlaps of instances, which may occur if the salient feature points lie too close. Any number of *non*-overlapping instances per path will be kept intact, however. For instance, the three serifs of a lowercase "m" can be replaced in one run.

Up to this stage, only the positions of the first and the last feature point of the search pattern on the path are known. Now the algorithm finds the points on the path closest to the appropriately transformed start point and end point of the search pattern. The path's segments are split here; the transformed segments of the search pattern are spliced in.

6. Prototype and results

The proposed method has been implemented as an extension for the Open Source vector illustration software Inkscape 0.46 (`http://www.inkscape.org/`) using Microsoft .NET. The user creates a path to be searched for and a path to replace it with as the two topmost items in the document and then invokes the extension. Inkscape passes the document as an SVG file (`http://www.w3.org/Graphics/SVG/`) to the external program, which eventually returns its result as another SVG file.

An illustration of nearly 900 paths (more than 600 characters of different typefaces) comprising about 40,000 Bézier segments in total requires nearly three minutes for the search & replace process on an 800 MHz notebook computer with a search pattern of 37 Bézier segments. The largest part of this time—more than two minutes—is spent on the matching phase, in particular the rendering into bitmaps.

7. Conclusion and outlook

This paper has presented a method to search for and replace parts of curves at arbitrary positions. It draws upon techniques ranging from finding salient points to comparing bitmap images. The method can act as a drastic time-saver for tasks that occur in graphics design. Nonetheless, a "live" preview would be helpful to indicate beforehand which parts of an illustration are changed in which way. This requires a speedup, for which there are many avenues: Several shapes can be handled in parallel on a multi-core processor; the validation through a bitmap can be executed with extreme performance on the graphics card, possibly using a stencil buffer and the pixel counters intended for occlusion culling; some parts of the algorithm could benefit from hashing, dynamic programming, and precomputation, e. g. of distance fields; the salient points can continuously be computed in the background while the user is editing the graphics document. Not only the speed but also the result can be improved: The bitmap test (see Figure 5) only checks for a tolerance zone; it could be improved by a later test that takes the order of the curve's points into account (Fréchet distance, see [SR02]); the insertion of the transformed pattern into the path needs to be fine-tuned for perfect transitions.

References

[FW94] FISCHLER M. A., WOLF H. C.: Locating perceptually salient points on planar curves. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 2 (1994), 113–129.

[HH93] HERTZ J., HERSCH R. D.: Analysing character shapes by string matching techniques. *Electronic Publishing 6* (1993), 261–272.

[HHGH98] HERTZ J., HU C., GONCZAROWSKI J., HERSCH R. D.: A window-based method for automatic typographic parameter extraction. In *Proc. EP '98/RIDT '98* (1998), pp. 44–54.

[HOCS02] HERTZMANN A., OLIVER N., CURLESS B., SEITZ S. M.: Curve analogies. In *Proc. Eurographics Workshop on Rendering '02* (2002), pp. 233–245.

[KB88] KURLANDER D., BIER E. A.: Graphical search and replace. *Computer Graphics 22* (1988), 113–120.

[KF92] KURLANDER D., FEINER S.: Interactive constraint-based search and replace. In *Proc. CHI '92* (1992), pp. 609–618.

[MBM05] MORI M.-G., BELONGIE M.-S., MALIK S. M.-J.: Efficient shape matching using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 27*, 11 (2005), 1832–1837.

[MHC*06] MANAY S., HONG B.-W., CREMERS D., YEZZI A. J., SOATTO S.: Integral invariants for shape matching. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 10 (2006), 1602–1618.

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L. J.: Discovering structural regularity in 3D geometry. *ACM TOG 27*, 3, Article 43 (2008).

[SR02] SAFONOVA A., ROSSIGNAC J.: Compressed piecewise-circular approximations of 3D curves. Georgia Institute of Technology, GIT-GVU-01-05, 2002.

[TV08] TANGELDER J. W., VELTKAMP R. C.: A survey of content based 3D shape retrieval methods. *Multimedia Tools Appl. 39*, 3 (2008), 441–471.

[VH01] VELTKAMP R. C., HAGEDOORN M.: State of the art in shape matching. In *Principles of visual information retrieval*, Lew M. S., (Ed.). Springer, London, UK, 2001, pp. 87–119.