Abstract

Harmonograms are visual designs based on harmonic analyses of strings of characters. Fourier descriptors are synthesized according to character positions and values, and then traditional methods are used to generate the resulting curves. Undersampling introduces visual artifacts that actually enhance the final designs. While their primary use is for the creation of artistic designs, harmonograms may have a practical application as a method for encoding and decoding information via the frequency domain.

1. Introduction

Just as a monogram consists of “two or more letters, especially a person’s initials, interwoven as a device” [1], a harmonic monogram or harmonogram is a design described by a harmonic series synthesized from a string of characters. The main difference between the two is that a monogram is based upon the component letter shapes, while a harmonogram is based upon the component letter positions and values.

Harmonograms should not be confused with harmonographs, which are physical devices for drawing curves using pens attached to oscillating pendula [2]. Harmonographs, invented in the 19th century, typically draw simple harmonic curves such as lissajous figures until friction brings them to a halt.

Fourier descriptors provide a convenient method for generating harmonograms. The following sections briefly introduce the principles behind Fourier descriptors, then go on to demonstrate their use in the synthetic generation of curves.

2. Fourier descriptors

Fourier descriptors are an invariant measurement for describing plane closed curves and distinguishing between different shapes. They were first suggested by R. Cosgriff in 1960, but the 1972 paper “Fourier Descriptors for Plane Closed Curves” by Zahn and Roskies [3] is generally regarded to be the seminal work in this area. In the original formulation, curves are represented as a function of arc length by the accumulated change in angle since their starting point, using a Fourier series.

For example, Fig. 1 shows how a circle can be represented by its accumulated change in angle. A starting point is chosen and its direction noted. The graph on the right is an angular profile that shows the current angle relative to the starting angle as the curve is traversed (dotted line). Note that the angular profile starts at $\theta = 0$ and finishes at $\theta = -2\pi$, as the total angular bend along a closed curve’s length must be $-2\pi$ if it is followed in the clockwise direction. In other words, following the curve will eventually bring you full circle. The solid line shows the curve’s normalized angular profile, which is given by adding the current position along the curve’s length (that is, the fraction of the curve that has been traversed) to the total angular bend. The normalized angular profile starts and ends at $\theta = 0$, hence is periodic.

Fig. 2 shows the angular profiles and normalized angular profiles of two other closed plane curves, by way of further example. In both cases the angular profiles again start at $\theta = 0$ and finish at $\theta = -2\pi$, and the normalized angular profiles again start and end at $\theta = 0$. The rightmost profile contains plateaus corresponding to linear segments of the curve and discontinuities corresponding to corners.
A closed plane curve of total length $L$ may be parameterized by its arc length $l$ where $0 \leq l \leq L$. A cumulative angular function $f(l)$ may then be defined which gives the net amount of angular bend at point $l$ relative to the starting angle. The start and end values of this function are

$$f(0) = 0, \quad (1)$$

$$f(L) = \frac{2\pi}{a_0}, \quad (2)$$

This cumulative angular bend function can be normalized to the time domain corresponding to the curve’s lifetime $t = [0..1]$ as follows:

$$\phi^*(t) = \phi\left(\frac{Lt}{2\pi}\right) + t. \quad (3)$$

Fig. 1. A circle, a starting point on its boundary, and its angular profile.

A curve’s Fourier descriptors essentially encode its angular profile as a harmonic series, from which the curve can be readily reconstructed. Fig. 3 (left) shows how sample points may be obtained at regular intervals along the curve’s path by finding the accumulated angle at each point.

**3. Curve reconstruction**

This allows all curves with identical shape and starting point to be uniquely defined by the periodic function $\phi^*(t)$. Curves have *identical shape* if they differ only by a combination of translation, scale and rotation [3].

Note that the normalized cumulative angular bend function $\phi^*(t)$ will be always be zero if the curve is a circle, since the circle’s angular bend changes constantly along its length. The function $\phi^*(t)$ therefore measures the way in which a curve differs from a circle.

The normalized cumulative angular bend function can be expanded into the frequency domain as a Fourier series. Eq. (4) shows the general form and Eq. (5) the more convenient polar form:

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt), \quad (4)$$

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} A_k \cos(kt - \alpha_k). \quad (5)$$

The curve is thus described by the Fourier descriptors $A$ and $\alpha$, where $A_k$ is the harmonic amplitude at the $k$th frequency, and $\alpha_k$ is the phase angle at the $k$th frequency.

Fourier descriptors have become a standard tool for practitioners of computer vision and pattern recognition. Various approaches now exist, based on the same basic principles but differing in detail and implementation. For example, the elliptic Fourier features of Kuhl and Giardina [4] are now widely used for classifying and comparing shapes.

**Fig. 2. Two closed planar shapes and their angular profiles.**
point, taking a unit step forward, then recalculating the accumulated angle for the next step.

These samples provide a polygonal approximation of the curve (Fig. 3, middle), which may then be smoothed using spline segments (right). A Catmull-Rom approach is used, meaning that the spline’s direction at each sample is parallel to a line between the previous sample and the next [5]. Curve reconstruction approaches 100% precision as the number of samples approaches infinity.

The starting point of the reconstructed curve is determined as follows:

\[
\mu_0 = \sum_{k=1}^{\infty} A_k \cos \alpha_k. \tag{6}
\]

Listing 1 shows C++ code for reconstructing a curve given its starting point and Fourier descriptor sets A (amplitude) and P (phase).

One disadvantage of Zahn and Roskies’ cumulative angle approach is that the shape of the reconstructed curve may differ significantly from its original (and may not even remain closed) if its Fourier descriptors are adjusted in the frequency domain. It might be desired, for instance, to suppress higher frequency harmonics in order to remove fine detail from the curve. Elliptic Fourier features are better at maintaining a curve’s overall shape and can guarantee closure despite operations on descriptors in the frequency domain. In addition, Zhang and Lu [6] found that descriptors based on curve position relative to the origin outperformed the cumulative angle approach for curve reconstruction, in a comparative study of various Fourier descriptor techniques.

However, we use Zahn and Roskies’ original approach in this paper due to its elegance and symmetrical properties, in addition to another surprising property (described in Section 6) that facilitates the synthetic generation of artistic curves.
4. Synthetic closed curve generation

Creating a fictitious set of Fourier descriptors and passing them to the curve reconstruction process allows the generation of synthetic curves. Zahn and Roskies point out two conditions under which the resulting curves are guaranteed to be closed:

1. $A_1$ is a zero of the first Bessel function $J_1(x)$ and all other $A_k = 0$.
2. $A_k = 0$ for all $k \mod m \neq 0$, where $k > 1$.

The first condition yields a limited set of less interesting curves and is beyond the scope of this paper, so the second condition will be used exclusively. This condition simply states that non-zero amplitude weights may only occur at frequencies with a common factor, for instance $A_2, A_4, A_6, A_8, \ldots$ or $A_3, A_6, A_9, A_{12}, \ldots$, and so on. This is achieved by specifying a modulus term and only creating fictitious weights for harmonics whose frequencies are a multiple of this modulus.

Fictitious phase values are created by selecting one of the following predefined phase types, and applying it to each harmonic with a non-zero weight: None (0), $\pi/2$, alternating ($3\pi/2, \pi/2, 3\pi/2, \pi/2, \ldots$), $k, k\pi/2, A_k$ and $A_k\pi/2$. This simple set of phase types proved sufficient for the creation of a variety of interesting shapes, although more complex mappings could of course be used.

5. Harmonograms

Harmonograms are created using fictitious Fourier descriptors based on the characters within a string. The user specifies a modulus value and phase type, then harmonic amplitude values are automatically determined based upon the ASCII values of each character (adjusted so that 'a' corresponds to an amplitude of 1) as follows:

$$A_{(c+1)m} = \text{ascii}(str[c]) - \text{ascii}(a') + 1,$$

where $str$ is the string, $c$ is the character’s position within the string, $m$ is the modulus value, and $\text{ascii()}$ is a function that returns the ASCII value of the character.

Listing 2 shows C++ code that converts a string to a set of harmonic amplitudes. The character 'a' = 1 is chosen as a convenient reference point near the center of the ASCII table ($\text{ascii('a')} = 97$ out of 255) which allows a similar number of positive and negative amplitudes. In addition, 'a' is easily remembered and makes the relative weightings of other characters obvious; lower case letters towards the start of the alphabet will yield low-frequency designs, while lowercase letters towards the end of the alphabet will yield high-frequency designs. With this conversion scheme, uppercase letters will yield negative amplitudes in the inverse relationship.

Any blank spaces within the string ($\text{ascii('')} = 32$) are treated as zero amplitudes for those frequencies. This is a convenient way of allowing sparse amplitude weightings while maintaining the readability of the string.

The information necessary for the reconstruction of a harmonogram is succinctly described by its string, modulus value $m$, and an additional scaling value $s$, in the form “string”$m,s$. This is called the harmonogram’s formal description in order to distinguish it from the harmonogram’s Fourier description. All non-zero amplitude values are multiplied by the constant scaling value $s$, which affects the complexity of the curve. Phase could also have been included in the formal description, but was deemed superfluous as phase was almost always set to either $\pi/2$ or Alternating.

```c++
void SynthesiseWeights
{
    CString str, // In: character string
    int mod, // In: frequency modulus
    float s, // In: scaling factor (default 1)
    float A[] // Out: harmonic amplitudes

    memset(A, 0, sizeof(A)); // initialize amplitudes to 0

    for (int c = 0; c < str.GetLength(); c++)
        if (str[c] != ' ')
            { int k = (c + 1) * mod; // frequency

                if (k < MAX_FREQUENCIES)
                    A[k] += s * (str[c] - 'a' + 1);
            }
}
```

Listing 2. C++ code for the creation of synthetic amplitude weights based on character positions and ASCII values.
Fig. 4 shows a number of harmonograms by way of example. Figs. 4(a) “e”_{2,1} and 4(b) “!”{2,1} show the basic harmonogram shape for single letters with modulus 2. The coils in Fig. 4(b) are more tightly wound as the character ‘!’ results in a greater absolute amplitude (63) than the character ‘e’ (5).

Figs. 4(c) “cb”_{4,1} and 4(d) “ab”{3,1} show more interesting designs resulting from longer character strings with higher moduli. Note that the first character in “ab” is a blank space, giving a zero value to the first harmonic candidate, and that the radial symmetry of the shapes corresponds directly to their modulus; $m = 3$ gives three-fold symmetry and $m = 4$ gives four-fold symmetry.

Fig. 4(e) shows a more elaborate harmonogram resulting from the description “pen and ink”_{2,1}. In general, the longer the string and the greater the modulus and absolute amplitude values, the more complex the design. Table 1 shows the actual amplitude values resulting from each of these formal descriptions.

Referring to Zahn and Roskies’ original paper [3], the formal descriptions “cb”_{4,1} and “ab”_{3,1} yield the same Fourier descriptors as their synthetic curves 8(d) and 8(e). It can be seen that the resulting shapes are identical in both papers, indicating that Zahn and Roskies’ approach has been implemented faithfully.

Fig. 5 further demonstrates the effect of the variables $m$ and $s$ by way of example. The top row shows harmonograms for the string “a” with $m = 2$ and $s = 1, 2, 3, 4$, the middle row shows the same string with $m = 3$ and $s = 1, 2, 3, 4$, and the bottom row shows the same string with $m = 4$ and $s = 1, 2, 3, 4$. It can be clearly seen that the modulus value $m$ affects the shape’s symmetry and the scale value $s$ affects the shape’s complexity.

We now consider an interesting property of harmonograms: that a curve’s generating character string can be reproduced from the curve itself using the reverse process, making harmonograms potentially useful as a method for encrypting and decrypting information. This might be achieved by applying a fast Fourier transform to a time series of sample points to determine the curve’s harmonic amplitudes, deducing the modulus and character positions from the distribution of non-zero amplitudes, and finally the ASCII values from the amplitudes themselves. The phase is immaterial in terms of string reproduction.

The encoding process would therefore be:

character string $\rightarrow$ synthesize Fourier descriptors $\rightarrow$ time series of sample points

and the decoding process would be:

time series of sample points $\rightarrow$ FFT $\rightarrow$ Fourier descriptors $\rightarrow$ character string

This reproducibility property would only hold when $s = 1$ (or was otherwise trivially deducible) and would only really be practical using a time series of sample points rather than an image of the curve itself, as more complex images become difficult to interpret. All of the examples in Fig. 4 have $s = 1$ and are therefore theoretically reproducible. However, it should be pointed out that this method of encryption would not be secure, and any encoded time series could be decoded by any person aware of the decoding process.

<table>
<thead>
<tr>
<th>Formal description</th>
<th>Amplitudes</th>
</tr>
</thead>
<tbody>
<tr>
<td>“e”_{2,1}</td>
<td>$A_2 = 5$</td>
</tr>
<tr>
<td>“!”_{2,1}</td>
<td>$A_2 = -63$</td>
</tr>
<tr>
<td>“cb”_{4,1}</td>
<td>$A_4 = 3$, $A_8 = 2$</td>
</tr>
<tr>
<td>“ab”_{3,1}</td>
<td>$A_6 = 1$, $A_2 = 2$</td>
</tr>
<tr>
<td>“pen and ink”_{2,1}</td>
<td>$A_2 = 16$, $A_4 = 5$, $A_8 = 14$, $A_{10} = 1$, $A_{12} = 14$, $A_{14} = 4$, $A_{18} = 9$, $A_{20} = 14$, $A_{22} = 11$</td>
</tr>
</tbody>
</table>
6. Undersampling

Fig. 6 demonstrates another interesting property of harmonograms alluded to previously. Fig. 6 (left) shows the harmonogram “$Z_{2,16}^2$” segmented by sample; note the similarity between this single-letter shape and that of Fig. 4(b). The curve is color-coded by parameter $t = [0..1]$ according to the periodic spectrum shown below the figure. This spectrum, based on a traversal of the Hue component of the HSV colour space, provides an attractive way of following the curve’s progress from start to end. The variable curve width is based on the local curvature at each sample.

Fig. 6 (left) was drawn using 2500 samples. This curve will retain the same basic shape for any greater number of
samples, except that the coils will converge more tightly for higher sample rates. However, the curve’s shape changes dramatically if fewer samples are taken. For instance, the middle and right figures show the curves resulting from the same formal description using 1250 and 1000 samples, respectively.

The baseline sample rate above which the curve remains stable depends on the string’s complexity, the modulus value \( m \), and the amplitude scale \( s \). Curves sampled below this rate are described as being \textit{undersampled}. The sample rate is an important parameter to the curve reconstruction function shown in Listing 1, however, it is intended more for artistic control of the resulting shape and is hence not included in a harmonogram’s formal description.

Undersampling effectively produces an abbreviated version of the original curve, however, the complexity of the shape and the periodic nature of the harmonic functions still allow interesting and closed results; indeed, most undersampled curves are more attractive than their fully sampled counterparts. An undersampled curve will generally be closed if the original curve is closed and the number of samples is a multiple of the frequency modulus \( m \).

It can be seen from the colour coding of Fig. 6 that the greater the sample rate, the more pronounced the curve’s coils and the more time the curve spends “down” those coils. Informally, undersampling has the general effect of opening up closed coils into attractive spiral vortices.

It is expected that undersampling, like the use of amplitude scales \( s > 1 \), would adversely affect the reproducibility of a harmonogram’s character string from its time series of samples, however, this has not yet been verified.

7. Examples

The remaining figures show some of the artistic possibilities of undersampling harmonograms. Figs. 7 and 8 show the harmonograms “What’s the buzz?”\textsuperscript{2,17} and “you”\textsuperscript{3,117} rendered using the periodic spectrum with variable curve width based on local curvature.

Fig. 9 again shows the harmonogram “you”\textsuperscript{3,117}, but at a higher sample rate and rendered as a polysphere using the POV-Ray ray tracer. Spheres were placed at regular intervals along the curve’s path with radius modulated by local curvature, then a periodic \( z \)-value oscillation was introduced to make the polysphere appear to weave over and under itself. Sphere depths were also adjusted based on radius so that larger spheres were placed behind smaller ones, allowing the many smaller curlicues to show through. This figure took 2.5 h to render at a resolution of 1000 \( \times \) 1000 pixels on a Pentium 4 3.2 GHz PC.

Even though Figs. 8 and 9 differ in detail, similar landmarks can be identified in each (for instance, the prominent loops midway along each of the three sides) since both curves derive from the same formal description.

Fig. 10 demonstrates how multiple harmonograms may be combined to form more complex and abstract designs. The lower-frequency component “a”\textsuperscript{3,141} dictates the curve’s overall shape, while the higher-frequency component “a”\textsuperscript{7,2102} creates finer periodic undulations along its path. The fact that neither of the moduli (3, 7) are factors of the sample rate (10,000) would account for its lack of symmetry. This would also mean that the curve is most likely not closed, although this is difficult to determine visually and does not make that much difference in this case anyway.
Fig. 8. A picture of “you”₃,₁₁₇ at 12,003 samples.

Fig. 9. The darker side of “you”₃,₁₁₇ at 14,004 samples (POV-Ray polysphere).
8. Conclusion

This paper introduces harmonograms and demonstrates their use in the creation of artistic designs. Future work might include further investigation into the relationship between sample rate, frequency modulus and curve closure, the conditions for which undersampled curves might be reproducible (if any), and their practical use as a method for encoding and decoding information.

A faster exploration of the harmonogram design space might be achieved using an evolutionary approach similar to that described by Sims [7] and Smith [8], in which the user “babysits” the evolutionary process by selecting those designs they prefer out of several candidates per generation.

It may also be interesting to investigate whether harmonograms could be used to detect harmonic linguistic features within a language, perhaps by using more complex conversion schemes (based on letter frequency, phonetic breakdown, vowel/consonant distribution, etc) on complete grammatical phrases. A comparative study across several languages may reveal harmonic features specific to particular languages and those common to all.

Thanks to Phil Bordelon for suggestions and corrections.

References