Stipple Tunes: An Artistic Form of Uncompressed Image in Audio Steganography

Alexa Lewis[1], Christopher J Tralie[1][0000−0003−4206−1963]

Ursinus College, Department of Mathematics And Computer Science, Collegeville, PA, USA

Abstract. We present an artistic audio steganography technique for hiding stipple images inside of uncompressed audio that we dub "stipple tunes." Given an audio carrier and a stipple image to hide, the goal is to manipulate samples in the left and right audio channels to draw the stipple points; that is, the left and right channels are interpreted, respectively, as X and Y coordinates in the Cartesian plane. To accomplish this, we devise an objective function that pans the audio and restricts samples to the stipple, while minimizing error, which we solve using the Viterbi algorithm. Decoding the hidden image is trivial; we simply create a scatterplot of the audio samples. We provide code, examples, and an interactive viewer in Javascript at `https://ctralie.github.io/StippleTunes/Viewer/`

1 Introduction

Steganography is the process of hiding one data stream "in plain sight" in another "carrier" data stream. In audio steganography [6][8], audio acts as a carrier. In any steganography technique, a simple scheme involves hiding data in the least significant bit of samples [5]. In 16-bit audio, this is inaudible. The downside of such techniques is that compression will destroy the hidden data, and it is easy to detect statistically [9]. However, in this work, we treat steganography as more of an artistic endeavor; we are concerned less with the data being compressed or statistically or audibly hidden, as long as the audio is still pleasant to listen to. To that end, we pursue an uncompressed audio steganography technique that we dub "stipple tunes," which is specifically designed to hide images in audio. Our goal is to spread an audio carrier across two channels in such a way that each pair of audio samples, when thought of as a point, plots an XY scatterplot that creates the hidden image. Figure 1 and Figure 2 show two examples.

Using the audio channels as coordinates on the Cartesian plane is spiritually similar to oscilloscope music [1][2][14], though we don't "connect the dots" as an oscilloscope would. It is also worth mentioning recent works that trained neural networks to hide full resolution color images [4][10][13][7] in audio, though we want our technique to be easier to explain and implement.

2 Stipple Tunes

To hide images in audio, we first turn to an intermediate representation: the *stipple pattern*, or a collection of dots that resembles the image. We use the technique of Secord [12] to automatically create stipples. This technique samples randomly from a density function that is higher in darker regions of the image, and then it moves the dots towards their Voronoi centers repeatedly (Lloyd's algorithm) until they converge to a more uniform, aesthetically pleasing distribution. To make sure our image picks up on important edges, even if they are brighter, we also make the density function higher in regions that are closer to edges, which we detect with a Canny edge detector [3].

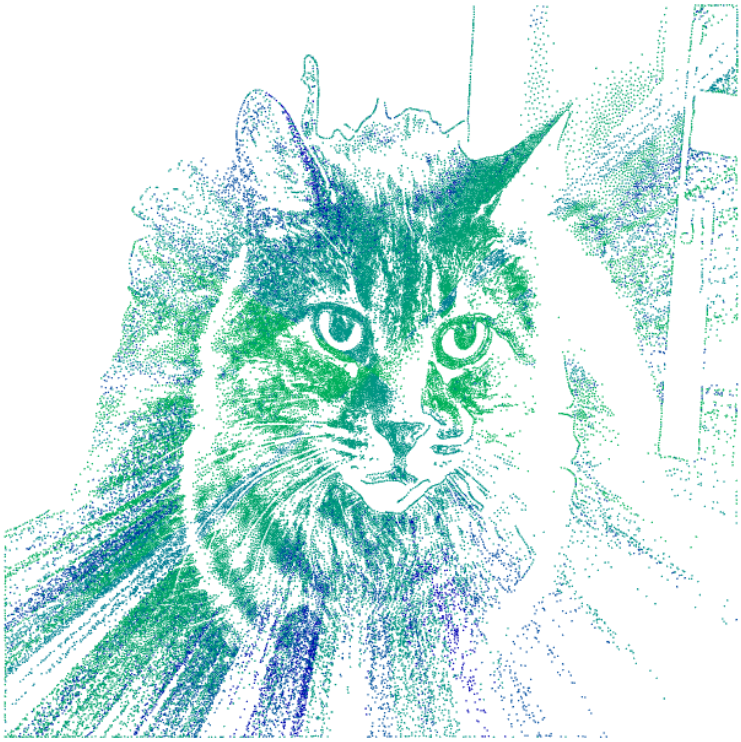

Fig. 1. A stipple tune on Layla the cat, using a 30 second clip from Eric Clapton's "Layla," created from a stipple with 100,000 points.

Once we have the stipple pattern and a single channel audio carrier $x[j]$, we turn the audio stream into a 2D curve by simply repeating the channel twice: one for each coordinate. From there, a simple idea is to find the nearest neighbor in the stipple pattern to each 2D audio point $(x[j], x[j])$. However, this has an immediate drawback since the curve simply moves back and forth along the line

Fig. 2. A stipple tune on a picture of Martin Luther King Jr., using a 30 second clip from his "I Have A Dream" speech.

$y = x$, so nearest neighbors would concentrate near this line. To encourage the algorithm to explore points away from this line, we slowly rotate the line and sweep the entire stipple. This corresponds to panning the audio between two channels which, while audible, is not unpleasant.

Formally, let $Y[j] = x[j] + ix[j]$ be an embedding of $(x[j], x[j])$ in the complex plane, and similarly embed the stipple Z in the complex plane. Then, we introduce a hidden state $\theta[j]$ so that we actually find the nearest neighbor from the points $Y_\theta[j] = Y[j]e^{i\theta[j]}$ to the stipple pattern Z. The effect of $\theta[j]$ is to pan between the left and right audio channels, and snapping $Y_\theta[j]$ to the nearest point in the stipple can be thought of an unusual form of quantization.

Crucially, we encourage the line to move and sweep the whole image by forcing $\theta[j+1] > \theta[j] + \epsilon$ for some $\epsilon > 0$. We can solve for the hidden states $\theta[j]$ using the Viterbi algorithm. Rather than maximizing a probability, as in the traditional application of Viterbi to HMMs, we seek to *minimize* the sum of nearest neighbor distances[1]. In this way, our application is similar in spirit to corpus-based concatenative synthesis [11], where the "corpus" is simply the stipple pattern. Algorithm 1 provides more details. In practice, we discretize θ by a factor *win* coarser than audio sample rate to keep the Viterbi algorithm tractable. We also discretize the possible rotation angles to *na*, and we force

[1] Of course, a probability can always be converted to a "distance" via a negative log

adjacent angle states to be between 1 and $tw < na$ of each other so that adjacent angles have to change, but not by an arbitrary amount. We also use a grid (with resolution equal to that of the image) to perform approximate nearest neighbor of points in Y to stipple points Z.

Figure 1 shows an example of mapping a stipple of a special cat named Layla to a 30 second clip from Eric Clapton's "Layla," using $na = 60$, $win = f_s = 44100$, and $tw = 10$. Since win is the sample rate f_s, we only have one state per second, but we find this is enough to get a good sweep through the stipple.

Algorithm 1 Stipple Tunes Algorithm

1: **procedure** StippleTune(Z, x, na, win, tw) ▷ Z is stipple, x is audio samples, na is number of discrete angle states, win is number of samples between angle states, and tw is amount by which angle can jump each step
2: $N \leftarrow \text{len}(x)$ ▷ Number of audio samples
3: $Y \leftarrow x + ix$
4: $M \leftarrow \text{ceil}(N/\text{win})$
5: $C[i, 0] \leftarrow 0, C[i, j > 1] \leftarrow \infty$ ▷ $na \times M$ Cumulative cost matrix
6: $I[i, j] \leftarrow 0$ ▷ $na \times M$ backpointers to best preceding state
7: **for** $t = 2 : M$ **do**
8: **for** $j = 1 : na$ **do**
9: $\theta_j \leftarrow 2\pi j/\text{na}$
10: **for** $k = j - \text{tw} : j - 1 \mod \text{na}$ **do**
11: $\theta_k \leftarrow 2\pi k/\text{na}$
12: Let $\theta_\ell \leftarrow \theta_j + (\theta_k - \theta_j)/\text{win}$
13: $d \leftarrow \sum_{\ell=1}^{\text{win}} dZ_{NN}(Y[\text{win} * t + \ell]e^{i\theta_\ell})$ ▷ Sum distances to the nearest neighbors in Z of all rotated Y points
14: **if** $C[k, t-1] + d < C[j, t]$ **then**
15: $C[j, t] \leftarrow C[k, t-1] + d$
16: $I[j, t] \leftarrow k$ ▷ Remember optimal transition
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21:
22: Backtrace I to obtain the optimal sequence of angle states
23: Linearly interpolate between each angle state (line 11) to compute $\theta[k], k = 1$ to N
24: Let X_k be the nearest neighbor in Z to $Y_k e^{i\theta[k]}$
25: **return** X
26: **end procedure**

3 Comparison To LSB Steganography

Let's suppose our audio is sampled at 44100hz. Then an LSB technique on 2 channel audio would transmit 88200 bits/second. Let's further suppose our

stipple has been discretized to a 1024x1024 grid, so that each stipple location requires 20 bits to transmit. This means that we could transmit 4410 stipple locations per second with an LSB technique, or 132,300 samples over a 30 second span. By contrast, with our technique, we could technically get a stipple location at every sample, for 10x as many locations per second. However, it is unlikely that Algorithm 1 would choose every stipple point if we used this many. In both Figure 1 and Figure 2, we used stipples with 100,000 samples over a period of 30 seconds, and most stipple samples were chosen, so this has a similar capacity to the LSB technique. However, our technique is incredibly easy to decode; we simply create a scatterplot of the 2 channel audio samples. This is also easier to explain to a non technical audience than LSB encoding.

References

1. Rendering shapes through audio signals. `https://felixonline.co.uk/issue/1773/science/rendering-shapes-through-audio-signals`, accessed: 2022-07-19
2. Ball, J.H.: Osci-render. `https://github.com/jameshball/osci-render` (2022), accessed: 2022-07-19
3. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6), 679–698 (1986)
4. Cui, W., Liu, S., Jiang, F., Liu, Y., Zhao, D.: Multi-stage residual hiding for image-into-audio steganography
5. Cvejic, N., Seppanen, T.: A wavelet domain LSB insertion algorithm for high capacity audio steganography. In: Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop. pp. 53–55. IEEE
6. Djebbar, F., Ayad, B., Meraim, K.A., Hamam, H.: Comparative study of digital audio steganography techniques **2012**(1), 25
7. Domènech Abelló, T.: Hiding images in their spoken narratives. Master's thesis, Universitat Politècnica de Catalunya (2022)
8. Dutta, H., Das, R.K., Nandi, S., Prasanna, S.R.M.: An overview of digital audio steganography **37**(6), 632–650
9. Fridrich, J., Goljan, M., Du, R.: Reliable detection of lsb steganography in color and grayscale images. In: Proceedings of the 2001 workshop on Multimedia and security: new challenges. pp. 27–30 (2001)
10. Geleta, M., Punti, C., McGuinness, K., Pons, J., Canton, C., Giro-i Nieto, X.: PixInWav: Residual steganography for hiding pixels in audio
11. Schwarz, D.: Corpus-based concatenative synthesis. IEEE signal processing magazine **24**(2), 92–104 (2007)
12. Secord, A.: Weighted voronoi stippling. In: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering. pp. 37–43 (2002)
13. Takahashi, N., Singh, M.K., Mitsufuji, Y.: Source mixing and separation robust audio steganography
14. Teschler, L.: Making pictures from sound on an oscilloscope. `https://www.testandmeasurementtips.com/making-pictures-from-sound-on-an-oscilloscope-faq/`, accessed: 2022-12-01