How `fiddle~` Works

Nathan Whetsell

Contents

1 Introduction **1**

2 Computing the DFT **1**
2.1 Initial Computation 2
2.2 Spectral Interpolation 2
2.3 Frequency-Domain Windowing 5
2.4 Possible Issues in `fiddle~`'s Implementation 5

3 Sinusoidal Peak Estimation **6**

4 The Likelihood Histogram **6**

5 Future Work **6**

1 Introduction

For my final project, I attempted (and am still attempting) to reverse engineer the `fiddle~` object often used in Max/MSP (but also available in Pure Data as well as a few obsolete languages). `fiddle~` is a tool used for realtime pitch tracking, that is, the estimation of the fundamental frequency of a sound with non-constant pitch.

2 Computing the DFT

The first step of a frequency-domain pitch tracker like `fiddle~` is, of course, to convert the input signal from a time-domain representation to a frequency-domain representation. This is typically done using the discrete Fourier transform (DFT). The DFT can be a computationally expensive undertaking if not performed with some forethought; fortunately, fast algorithms for implementing the DFT exist, among them the ubiquitous Fast Fourier Transform (FFT) (Cooley and Tukey 1965). Incredibly, `fiddle~` does not employ the FFT to compute a DFT.

The algorithm used to compute the DFT appears to be a highly optimized version of the discrete Hartley transform (DHT). A complete description of this transform is beyond the scope of this final project—any reasonably fast DFT implementation will suffice for `fiddle~`'s purposes—but it is worth saying a few words about it. The mathematical foundations of the DHT were first described in (Hartley 1942), but the connection to the DFT (and FFT) was not recognized until much later, in (Bracewell 1983). In this paper, Bracewell also described a fast Hartley transform (FHT), which was the subject of United States patent number 4,646,256, issued to Stanford University. (Interestingly, the patent itself calls it the "discrete Bracewell transform.") In 1994, this patent was placed in the public domain (Bracewell 1995).

The particular implementation of the DHT used in `fiddle~` appears to be based on code written by one Ron Mayer sometime in 1993. Surprisingly, the primary reference describing this code—(Mayer 1993)—refers to an Internet message board post (in fact a copy of a message board post). It appears that in many applications the "Mayer FHT" has been superseded by modern algorithms. Mayer himself suggests the whimsically titled Fastest Fourier Transform in the West (FFTW) for general purpose applications (Mayer).

Let us now work out how `fiddle~` computes a DFT. In essence, `fiddle~` computes a DFT of a block of input, zeropadded by a factor of four. There are three main steps—initial computation, spectral interpolation, and frequency-domain windowing—that we will consider in detail. (For the reader familiar with (Puckette, Apel, and Zicarelli 1998), this is a complete description of the "interesting trick" referenced in that work.)

2.1 Initial Computation

Suppose we have a real, length N signal $x[n]$, and we modulate this signal by the complex exponential $e^{-j\frac{\pi}{2N}n}$ to obtain

$$x_{\text{modulate}}[n] = x[n]e^{-j\frac{\pi}{2N}n}. \tag{1}$$

We can compute the DFT of this modulated signal as

$$X_{\text{modulate}}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}n}e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+1)n} \tag{2}$$

for $k \in [0, N-1]$. Now, suppose we take the same signal $x[n]$, and rather than modulate it, we zeropad it to length $4N$ to obtain a signal $x_{\text{zeropad}}[n]$. The DFT of this new signal is

$$X_{\text{zeropad}}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{4N}kn} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}kn} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}kn} \tag{3}$$

for $k \in [0, 4N-1]$. Note that

$$X_{\text{modulate}}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+1)n} = X_{\text{zeropad}}[4k+1] \tag{4}$$

and

$$X^*_{\text{modulate}}[N-k-1] = \sum_{n=0}^{N-1} x[n]e^{j\frac{\pi}{2N}[4(N-k-1)+1]n} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+3)n} = X_{\text{zeropad}}[4k+3] \tag{5}$$

for $k \in \left[0, \frac{N}{2}\right]$. This implies that we can compute the spectrum in the odd-numbered DFT bins of X_{zeropad} using only X_{modulate}, which is a quarter of the length of X_{zeropad}. To obtain the spectrum in the even-numbered DFT bins, we must interpolate this spectrum from the odd-numbered bins.

2.2 Spectral Interpolation

In general, the length N DFT at bin k of a length $M \leq N$ signal $x[n]$ is given by

$$X[k] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn}.$$

Suppose we have

$$X[k-\ell_i] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k-\ell_i)n}$$
$$X[k+\ell_i] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k+\ell_i)n}$$

where the ℓ_i's are positive integers. Note that

$$\begin{aligned} X[k-\ell_i] + X[k+\ell_i] &= \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}(k-\ell_i)n} + \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}(k+\ell_i)n} \\ &= \sum_{n=0}^{M-1} x[n] \left(e^{-j\frac{2\pi}{N}(k-\ell_i)n} + e^{-j\frac{2\pi}{N}(k+\ell_i)n} \right) \\ &= 2 \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}kn} \cos\left(\frac{2\pi}{N} \ell_i n \right). \end{aligned}$$

Summing over the ℓ_i's, we have

$$\begin{aligned} \sum_{i=1}^{L} X[k-\ell_i] + X[k+\ell_i] &= 2 \sum_{i=1}^{L} \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}kn} \cos\left(\frac{2\pi}{N} \ell_i n \right) \\ &= 2 \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}kn} \sum_{i=1}^{L} \cos\left(\frac{2\pi}{N} \ell_i n \right) \end{aligned}$$

Finally, if

$$\sum_{i=1}^{L} a_i \cos\left(\frac{2\pi}{N} \ell_i n \right) = \frac{1}{2} \tag{6}$$

for $n \in [0, M-1]$ and some set of a_i's, then

$$\sum_{i=1}^{L} a_i (X[k-\ell_i] + X[k+\ell_i]) = \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{N}kn} = X[k]. \tag{7}$$

This is promising. In words, we can compute a DFT value $X[k]$ as a weighted sum—a superposition—of the DFT values on either side of $X[k]$. Also, we made no assumptions about what the ℓ_i's are; we said nothing about *which* bins on either side of $X[k]$ we need to look at. Thus, choosing just the odd-numbered bins should present no difficulties.

There are, unfortunately, two problems. First, in general, there is *no* set of a_i's that makes equation (6) hold for a set of ℓ_i's. We will find that this problem is not particularly troublesome; we can estimate a set of a_i's that makes equation (6) "almost hold" in an optimal sense. A second, and much more subtle, problem arises if all the ℓ_i's are odd (and they will be in `fiddle~`).

Fig. 1 shows several cosines of the form

$$\cos\left(\frac{2\pi}{2048} \ell_i n \right),$$

where the ℓ_i's are all odd. These are the cosines we'll be working with if we perform interpolation for a length $N = 2048$ DFT. For equation (6) to hold, these cosines must sum up to $\frac{1}{2}$ for some set of weightings, in some region $[0, M-1]$. Now, suppose that the signal length $M = 512$. We now have a problem; *all* of the cosines equal 0 at $n = 512$. No matter what weightings we use, the point at $n = 512$ can never equal $\frac{1}{2}$. Clearly, points near $n = 512$ will have values close to 0. Near $n = 512$, there is absolutely no hope of equation (6) holding.

What can we do about this? If we delay each cosine by $\frac{\pi}{4}\ell_i$, equation (6) becomes

$$\sum_{i=1}^{L} a_i \cos\left[\left(\frac{2\pi}{N} n - \frac{\pi}{4} \right) \ell_i \right] = \frac{1}{2}. \tag{8}$$

These shifted cosines are plotted in Fig. 2. We should have a much easier time finding weights that make the cosines in [0,511] approximate $\frac{1}{2}$. This phase shift can be implemented by modifying equation (7) to

$$\sum_{i=1}^{L} a_i \left(X[k-\ell_i] e^{-j\frac{\pi}{4}\ell_i} + X[k+\ell_i] e^{j\frac{\pi}{4}\ell_i} \right) = X[k]. \tag{9}$$

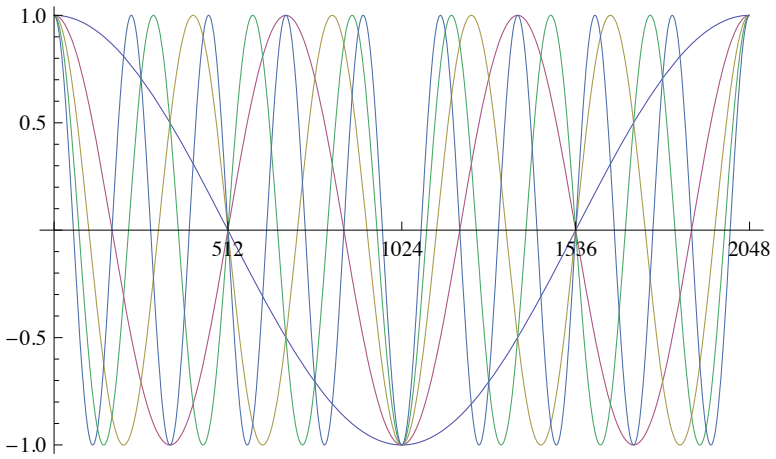

Figure 1: Cosines $\cos\left(\frac{2\pi}{N}\ell_i n\right)$, with ℓ_i odd.

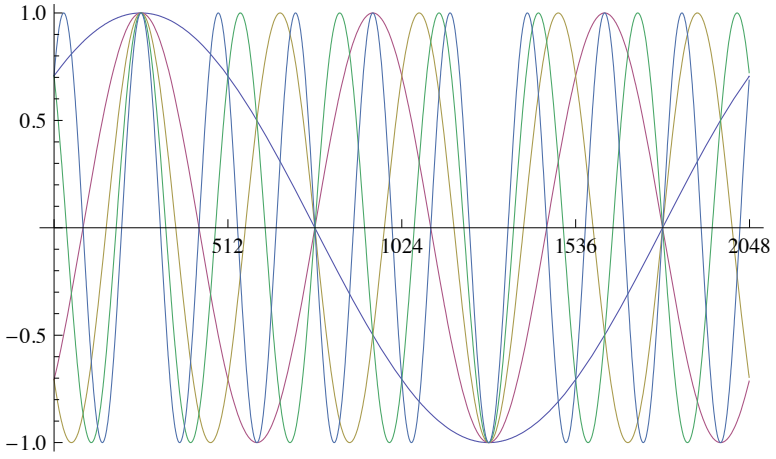

Figure 2: Cosines $\cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)\ell_i\right]$, with ℓ_i odd.

We are now ready to approximate a set of a_i's that "nearly" conforms to equation (8). Define

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_L \end{bmatrix}^T$$
$$\mathbf{b}(n) = \begin{bmatrix} \cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)\ell_1\right] & \cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)\ell_2\right] & \cdots & \cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)\ell_L\right] \end{bmatrix}^T.$$

Also define the error E as

$$E = \sum_{n=0}^{M-1}\left[\mathbf{a}^T\mathbf{b}(n) - \frac{1}{2}\right]^2 = \mathbf{a}^T\left[\sum_{n=0}^{M-1}\mathbf{b}(n)\mathbf{b}^T(n)\right]\mathbf{a} - \mathbf{a}^T\sum_{n=0}^{M-1}\mathbf{b}(n) + \frac{1}{4}. \tag{10}$$

Taking the derivative of equation (10) with respect to $\mathbf{a}$ and setting it equal to 0, we obtain

$$2\left[\sum_{n=0}^{M-1}\mathbf{b}(n)\mathbf{b}^T(n)\right]\mathbf{a} - \sum_{n=0}^{M-1}\mathbf{b}(n) = 0 \Rightarrow \mathbf{a} = \frac{1}{2}\left[\sum_{n=0}^{M-1}\mathbf{b}(n)\mathbf{b}^T(n)\right]^{-1}\sum_{n=0}^{M-1}\mathbf{b}(n). \tag{11}$$

For the purposes of the `fiddle~` object, the appropriate parameter values are $M = 512$, $N = 2048$, and $\{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5\} = \{1, 3, 5, 7, 9\}$. Computing the optimal $\mathbf{a}$ for these parameters using equation (11) gives

$$\mathbf{a} = \begin{bmatrix} 0.614965 & -0.154600 & 0.051131 & -0.013506 & 0.002033 \end{bmatrix}^T.$$

The coefficients from the `fiddle~` source code are

$$\mathbf{a} = \begin{bmatrix} 0.613527 & -0.151193 & 0.047663 & -0.011374 & 0.001267 \end{bmatrix}^T$$

and are in reasonably close agreement. (The coefficients found by least-squares optimization give improved performance over the coefficients in the `fiddle~` source code, which were found "by trial and error.")

2.3 Frequency-Domain Windowing

The final step in `fiddle~`'s DFT computation is windowing the original block of audio. However, instead of multiplying the time-domain signal by a window, we convolve in the frequency domain.

If we are given a rectangular-windowed spectrum $X[k]$ that has been zeropadded by a factor of c, we can find its Hann-windowed version $X_H[k]$ using

$$X_H[k] = \frac{X[k]}{2} - \frac{X[k-c] + X[k+c]}{4} = (W_H * X)[k], \tag{12}$$

where $W_H = \left[-\frac{1}{4}, 0, \ldots, 0, \frac{1}{2}, 0, \ldots, 0, -\frac{1}{4}\right]$. To see this, write

$$\mathrm{IDFT}\{W_H[k]\} = \sum_{k=-c}^{c} W_H[k]e^{j\frac{2\pi}{M}kn} = -\frac{1}{4}e^{-j\frac{2\pi}{M}cn} + \frac{1}{2} - \frac{1}{4}e^{j\frac{2\pi}{M}cn} = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi}{M}cn\right). \tag{13}$$

Indeed, this is a (causal) Hann window for use with a signal zeropadded by a factor of c.

2.4 Possible Issues in `fiddle~`'s Implementation

In a certain sense, this is a description of the *algorithm* `fiddle~` employs to compute the DFT. The actual implementation is... idiosyncratic.

Before the frequency-domain windowing, various phase shifts are applied—seemingly inadvertently—to the spectrum. Bins $k = 1, 5, 9, \ldots$ are modulated by $e^{-j\frac{\pi}{2}}$, bins $k = 2, 6, 10, \ldots$ are modulated by $e^{-j\pi}$, and bins $k = 3, 7, 11, \ldots$ are modulated by $e^{j\frac{\pi}{2}}$.

The frequency-domain windowing is also a bit strange. If `fiddle~` zeropads a block of input by a factor of four (and it does), we should expect the Hann window to be implemented as

$$w_H[n] = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{8\pi}{M}n\right).$$

The Hann window actually employed assumes a zeropadding factor of two, and includes an additional gain:

$$w_H[n] = 1 - \cos\left(\frac{4\pi}{M}n\right).$$

3 Sinusoidal Peak Estimation

The next step in the `fiddle~` pitch-tracking algorithm is to estimate the frequencies of prominent spectral peaks. There are three sources describing precisely how `fiddle~` estimates the frequency of a spectral peak: (Puckette, Apel, and Zicarelli 1998), (Puckette and Brown 1998), and the `fiddle~` code. Each source gives a different equation. It is my *guess* that this equation from (Puckette and Brown 1998) is the one intended:

$$\omega_{\text{est}} = \frac{2\pi}{N}\left(k + \Re\left\{\frac{\beta(X[k-1] - X[k+1]) - 2\gamma(X[k-2] - X[k+2])}{2\alpha X[k] - \beta(X[k-1] + X[k+1]) + 2\gamma(X[k-2] + X[k+2])}\right\}\right). \tag{14}$$

The parameters α, β, and γ in equation (14) are terms of the Blackman window family, which we can define as

$$w_B[n] = w_R[n]\left[\alpha - \beta\cos\left(\frac{2\pi}{N}n\right) + \gamma\cos\left(\frac{4\pi}{N}n\right)\right], \tag{15}$$

where $w_R[n]$ is a rectangular window. This definition differs slightly from the one given in (Puckette and Brown 1998), but it is unclear what effect this difference has on equation (14). For a causal Hann window, we have $\{\alpha, \beta, \gamma\} = \{\frac{1}{2}, \frac{1}{2}, 0\}$, and equation (14) becomes

$$\omega_{\text{est}} = \frac{\pi}{N}\left(k + \Re\left\{\frac{X[k-1] - X[k+1]}{2X[k] - X[k-1] - X[k+1]}\right\}\right). \tag{16}$$

The equation given in (Puckette, Apel, and Zicarelli 1998) is

$$\omega_{\text{est}} = \frac{\pi}{N}\left(k + \Re\left\{\frac{X[k-2] - X[k+2]}{2X[k] - X[k-2] - X[k+2]}\right\}\right), \tag{17}$$

which corresponds to a Blackman family window with parameters $\{\alpha, \beta, \gamma\} = \{1, 0, -\frac{1}{4}\}$, an unusual window indeed. The `fiddle~` source code uses

$$\omega_{\text{est}} = \frac{\pi}{N}\left(k + \frac{\Re\{(X^*[k-2] - X^*[k+2])(2X[k] - X[k-2] - X[k+2])\}}{2|X[k]|}\right), \tag{18}$$

which seems somewhat out of left field.

4 The Likelihood Histogram

The next—and possibly most important—step is the computation of the likelihood histogram. It is this data structure that performs the "magic" of finding which spectral peaks correspond to harmonic pitches. This data structure appears to be the result of much experimentation, and untangling the C code to determine what exactly it is doing is an ongoing project.

5 Future Work

The final goal of this project is to port the `fiddle~` algorithm to MATLAB in order to compare `fiddle~` to the YIN algorithm described in (de Cheveigné and Kawahara 2002). The YIN algorithm already exists as MATLAB code; porting `fiddle~` to MATLAB will facilitate automated and very detailed testing.

For a test suite, I plan to use the publicly available sound databases described in (Camacho 2007).

References

Bracewell, R. N. 1983. Discrete Hartley transform. *Journal of the Optical Society of America.* 73(12):1832–5.

Bracewell, R. N. 1995. Computing with the Hartley transform. *Computers in Physics.* 9(4):373–9.

Camacho, A. 2007. SWIPE: A sawtooth waveform inspired pitch estimator for speech and music. Ph.D. thesis. University of Florida. Gainesville, FL, USA.

de Cheveigné, A., and H. Kawahara. 2002. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*. 111(4):1917–30.

Cooley, J. W., and J. W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*. 19(90):297–301.

Mayer, R. Performance and accuracy benchmarking for FFT (Fast Fourier Transform), RFFT (real-valued FFT) and FHT (Fast Hartley Transform) algorithms. http://www.geocities.com/ResearchTriangle/8869/fft_summary.html (accessed April 21, 2008).

Mayer, R. 1993. Real FFT comparison. http://www.geocities.com/ResearchTriangle/8869/1993_fft_summary.html (accessed April 21, 2008). Online copy of newsgroup post.

Hartley, R. V. L. 1942. A more symmetrical Fourier analysis applied to transmission problems. *Proceedings of the Institute of Radio Engineers*. 30(3):144–50.

Puckette, M. S., T. Apel, and D. D. Zicarelli. 1998. Real-time audio analysis tools for Pd and MSP. In *Proceedings of the International Computer Music Conference*. 109–12.

Puckette, M. S., and J. C. Brown. 1998. Accuracy of frequency estimates using the phase vocoder. *IEEE Transactions on Speech and Audio Processing*. 6(2):166–76.

Bibliography

Boersma, P. 1993. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. *Proceedings of the Institute of Phonetic Sciences 17*. 97–110.

Brown, J. C., and M. S. Puckette. 1993. A high resolution fundamental frequency determination based on phase changes of the Fourier transform. *Journal of the Acoustical Society of America*. 94(2):662–7.

Cooper, D., and K. C. Ng. 1996. A monophonic pitch-tracking algorithm based on waveform periodicity determinations using landmark points. *Computer Music Journal*. 20(3):70–8.

de la Cuadra, P., A. Master, and C. Sapp. 2001. Efficient pitch detection techniques for interactive music. In *Proceedings of the International Computer Music Conference* (ICMC 2001). 403–6.

Jacovitti, G., and G. Scarano. 1993. Discrete time techniques for time delay estimation. *IEEE Transactions on Signal Processing*. 41(2):525–33.

Kuhn, W. B. 1990. A real-time pitch recognition algorithm for music applications. *Computer Music Journal*. 14(3):60–71.

Lane, J. E. 1990. Pitch detection using a tunable IIR filter. *Computer Music Journal*. 14(3):46–59.

Licklider, J. C. R. 1951. A duplex theory of pitch perception. *Experientia*. 7(4):128–34.

Liu, J., T. Zheng, J. Deng, and W. Wu. 2005. Real-time pitch tracking based on combined SMDSF. In *Proceedings of the 9th European Conference on Speech Communication and Technology* (EUROSPEECH 2005). 301–4.

Marchand, S. 2001. An efficient pitch-tracking algorithm using a combination of Fourier transforms. In *Proceedings of the COST G-6 Conference on Digital Audio Effects* (DAFx-01). 170–4.

Markel, J. D. 1972. The SIFT algorithm for fundamental frequency estimation. *IEEE Transactions on Audio and Electroacoustics*. 20(5):367–77.

McGonegal, C. A., L. Rabiner, and A. E. Rosenberg. 1975. A semiautomatic pitch detector (SAPD). *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 23(6):570–4.

McLeod, P., and G. Wyvill. 2005. A smarter way to find pitch. In *Proceedings of the International Computer Music Conference*. 138–41.

Noll, A. M. 1966. Cepstrum pitch determination. *Journal of the Acoustical Society of America*. 41(2):293–309.

Noll, A. M. 1970. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum and a maximum likelihood estimate. In *Proceedings of the Symposium on Computer Processing in Communications*, vol. 19. Brooklyn: Polytechnic Press. 779-97.

Rabiner, L. R. 1977. On the use of autocorrelation analysis for pitch detection. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25(1):24–33.

Rabiner, L. R., M. J. Cheng, A. E. Rosenberg, and C. A. McGonegal. 1976. A comparative performance study of several pitch detection algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24(5):399–418.

Ross, J. R., H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley. 1974. Average magnitude difference function pitch extractor. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 22(5):353–62.