Feature Extraction for ASR: Pitch

Wantee Wang

2015-03-14 16:55:51 +0800

Contents

Pitch is a perceptual property that allows the ordering of sounds on a frequency-related scale. Pitch may be quantified as a frequency, which is referred as *Fundamental frequency* (F0). Pitch and pitch changes in words form the *tone* of a tonal language, such as Chinese.

RAPT[1] is a time-domain algorithm for F0 estimation, which Kaldi's pitch feature [2] is based on.

Most F0 estimators consist of three main steps:

1. Pre-processing, including DC-removing and framing, etc.;
2. Candidate generating, main method to estimate the F0;
3. Post-processing, choosing the best candidate and refining the result output.

Here, we present main ideas for RAPT algorithm and its variant introduced by Kaldi.

1 Cross-correlation and Autocorrelation

In signal processing, cross-correlation is a measure of similarity of two series as a function of the lag of one relative to the other.

For continuous functions f and g, the cross-correlation is defined as,

$$(f \star g)(\tau) = f(t) \star g(t) = \int_{-\infty}^{+\infty} f(t)g(\tau + t)dt$$

and the discrete form is,

$$(x \star y)[m] = x[n] \star y[n] = \sum_{n=-\infty}^{+\infty} x[n]y[m + n] \tag{1}$$

Here m is called the lag. The cross-correlation is similar in nature to the convolution of two functions. Recall that convolution of two function f and g is,

$$(f * g)(\tau) = f(t) * g(t) = \int_{-\infty}^{+\infty} f(t)g(\tau - t)dt$$

and

$$(x * y)[m] = x[n] * y[n] = \sum_{n=-\infty}^{+\infty} x[n]y[m - n] \tag{2}$$

It is easy to see $f(t) * g(t) = f(t) \star g(t)$.

Autocorrelation is defined as the cross-correlation of a signal with itself. For autocorrelation, there will always be a peak at a lag of zero.

From the definitions, we can compute a convolution of f and g by following steps:

1. Reflect functions g: $g(t) \rightarrow g(-t)$.
2. Add a time-offset, τ, which allows $g(\tau - t)$ to slide along the t-axis.
3. Find the integral of their product, wherever the two functions intersect.

Similar procedure can be performed for cross-correlation, i.e. just skipping the first step.

This figure (Figure 1) from Wikipedia illustrates the idea,

Autocorrelation can be used to find repeating patterns, or the period of a signal. For a periodic signal, its autocorrelation will have peaks at its period. This property can be used to find F0s.

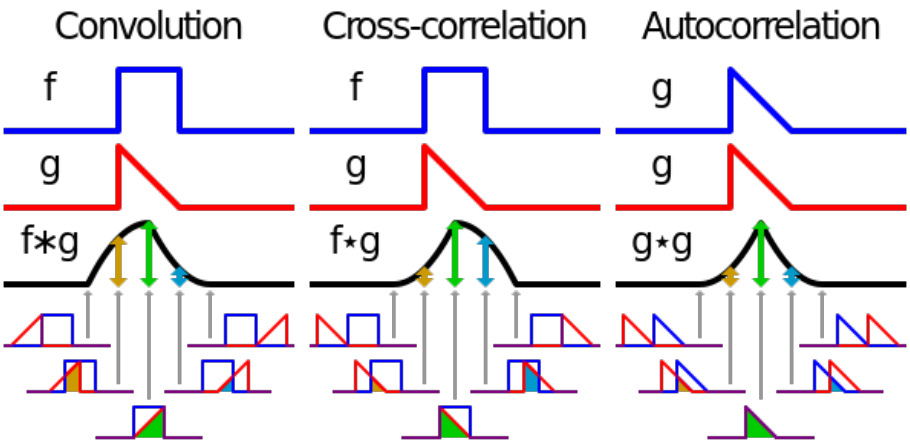

Figure 1: Visual comparison of convolution, cross-correlation and autocorrelation.

2 Normalized Cross-Correlation Function

In real signal processing, we would have to split the waveform into shorter frames. We perform the mathematical operations within frames, this leads to the *AutoCorrelation Function* (ACF), *Cross-Correlation Funcion* (CCF) and *Normalized Cross-Correlation Function* (NCCF).

Suppose the window size is N samples, for a frame i, the ACF of signal $s[n]$ is,

$$R_i[m] = \sum_{n=b_i}^{b_i+N-1-m} s[n]s[m+n] \tag{3}$$

And the CCF is,

$$\chi_i[m] = \sum_{n=b_i}^{b_i+N-1} s[n]s[m+n] \tag{4}$$

where b_i is the beginning sample of frame i.

At first glance, the equation for CCF (4) seems more likely to the autocorrelation formula. Notice that the difference between ACF and CCF is we clip the value of signal outside the window to zero when computing ACF. But as we computing CCF, we use the origin signal, thus we actually do not using the same signal to compute CCF, i.e., $x[n] = s[b_i : b_i+N-1]$ and $y[n+m] = s[b_i+m : b_i+m+N-1]$.

This slide uses many plots to show the ideas.

The drawback of the ACF is a stepwise "shortening" of the segment, however, the CCF also has problem: the shifted signal has much higher energy, and has no bound on values.

Thus we use normalization to compensate the energy, which leads to NCCF,

$$\phi_i[m] = \frac{\sum_{n=b_i}^{b_i+N-1} s[n]s[m+n]}{\sqrt{E[b_i]E[m+b_i]}} \tag{5}$$

where, $E[j]$ is energy for signal $s[j : j+N-1]$,

$$E[j] = \sum_{n=j}^{j+N-1} s^2[n]$$

Note that $-1.0 \leq \phi \leq 1.0$, ϕ tends to be close to 1.0 for lags corresponding to integer multiples of the "true" period.

3 RAPT

With NCCF, we can estimate the F0. To reduce the computation of NCCF, RAPT uses a two-pass NCCF procedure. The outline of RAPT is:

1. Down-sample the original signal;
2. Compute the NCCF of the low sample rate signal. Record the locations of local maxima;
3. Compute the NCCF of the original signal only in the vicinity of promising peaks found in the first pass. Every frame may have several candidates;
4. Find the best candidate for every frame using dynamic programming.

In order to form the DP recursion, we define the *local cost* for proposing F0 of frame i is $C_{i,j}$ is $d_{i,j}$, where $C_{i,j}$ is the value of j-th local maximum in ϕ at frame i. And the inter-frame F0 *transition cost* is $\delta_{i,j,k}$, which denotes the cost at frame i when hypotheses j and k at the current and previous frame. Thus, the total cost for j candidate at frame i is,

$$D_{i,j} = d_{i,j} + min_{k \in I_{i-1}}\{D_{i-1,k} + \delta_{i,j,k}\}, 1 \leq j \leq I_i,$$

with the initial condition

$$D_{0,j} = 0, 1 \leq j \leq I_0; I_0 = 2.$$

where, I_i denotes the number of candidates proposed at frame i, which is the number of non-zero-lag local maxima plus one (for unvoiced state).

Next, we need to choose a proper $d_{i,j}$ and a $\delta_{i,j,k}$[1].

Apparently, we should choose larger value of ϕ, i.e., $d_{i,j} \propto -C_{i,j}$. On the other side, in order to encourage the selection of shorter period, suppose $L_{i,j}$ is the corresponding lag of $C_{i,j}$, then $d_{i,j} \propto L_{i,j}$. Finally, we have,

$$d_{i,j} = 1 - C_{i,j}(1 - \beta L_{i,j})$$

where β is the factor to weight the penalization of long lag.

The main concern of the transition cost is to smooth the final F0 among frames, therefore we can penalty the changes of lags, i.e.,

$$\delta_{i,j,k} = \alpha \left| \log \frac{L_{i,j}}{L_{i-1,k}} \right|$$

similarly, α is a constant factor to weight the penalization.

4 Kaldi Pitch Tracker

Kaldi uses a highly modified version of the RAPT algorithm. The most difference is that Kaldi does not make a hard decision whether any given frame is voiced or unvoiced; instead, it assigns a pitch even to unvoiced frames while constraining the pitch trajectory to be continuous. And it is also produces a probability of voicing (POV) and other features used in ASR tasks.

References

[1] David Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[2] Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur. A pitch extraction algorithm tuned for automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2494–2498. IEEE, 2014.

[1] Note that $d_{i,j}$ and $\delta_{i,j,k}$ in the original RAPT paper looks like something more complicated than this post. Here, we just show the basic ideas in simpler form.