Optimization report

Bas Cornelissen (1006921)

March 29, 2016

1 Introduction

A lot of STS research highlighted the importance of pitch. To test whether pitch is indeed the main factor in the illusion, we look for a method that predicts the STS effect, based on pitch. The idea is simple: if a stimulus transforms to song, it has a pitch structure that easily allows for a musical interpretation. So we have to find a way to measure how easily you can interpret a given stimulus as song. The first step, then, is to extract the musical pitch structure.

Standard methods for pitch extraction from audio are available (e.g. autocorrelation, cross-correlation, or more recently pYIN). Getting from pitch contours to musical notes, is a problem that has not been studied in much depth. In the musical information retrieval world, this problem is known as music transcription. Most studies deal with the transcription of song lines in polyphonic music, which is much harder than transcription in monophonic music. Paradoxically, the harder problem attracted more attention and has been studied more extensively. Observing this hiatus Mauch et al. built a program called *Tony* that one the one hand tries to extract the tonal structure from an audio file, but also provides a neat interface to annotation the audio manually. The pitch data is extracted using pYIN, which they too developed, and to get to a symbolic representation, they use a hidden Markov model.

Although this model works fine on song data (according to the paper), it does not do a very good job on speech data, which is what we are concerned with. Part of the explanation might be that their model tries to detect stable tonal targets, and those are rare in ordinary speech. Our task is not to detect stable tonal targets, but to estimate which pitch you will hear in unstable tonal targets. (Of course, we know that the very stability already predicts the STS, but not the resulting melody.) This problem is a bit more obscure and I am not aware of previous studies tackling this problem. Therefore, we start at the very beginning.

So where are we to hear pitch anyway? The starting point is that roughly every syllable will be heard on a pitch after some repetitions. In fact, we can say a bit more, since syllables have some structure: they consist of various different kinds of sound. Of most important here is what phoneticians call the *nucleus*, which roughly coincides with the vowel in the syllable. This is the part of the syllable that has the clearest pitch. Our tactic is thus to estimate the pitch of the syllable nuclei and then possibly adjust those to the most likely musical structure. That really divides in three steps: first, to extract syllable nuclei, second to estimate their pitch and third, to map them to a likely musical structure.

Although we can solve the first problem by manually annotating the syllables, it would be desirable to be able to detect them automatically. That would enable us, for example, to automatically *identify* in a given speech fragment, the part that will transform to a song. We look at automatic nucleus detection first.

2 Nucleus detection

As far as I know, there is only one study that attempted to find an algorithm for nucleus detection,[1] a study by Jong and Wempe. They developed the algorithm to automatically estimate the *speech rate* of speech audio: the number of syllables per second, that is. So in a way, this problem is closely related to the problem of syllable segmentation, in which you have to identify the syllable boundaries. However, that problem is much harder since you for example have to separate consonants at syllable boundaries. Partly for that reason, I imagine, they focussed on the nuclei[2]

It turns out that nuclei are relatively easy to find. They correspond to peaks in the intensity curve of the audio. The algorithm proposed by Jong and Wempe essentially selected the peaks that were 'clear enough' as the nuclei. More formally, list the non-silent peaks with a pitch — those are the only serious candidates for nuclei — as $p_1, \ldots, p_n$. Let v_i be the valley or minimum after p_i. Moreover write $I(t)$ for the intensity at time t such that $I(p_3)$ is the intensity of the third peak (in dB). The nuclei are then the peaks p_i for which the following dip $|p_i - v_i|$ is deep enough, i.e. above some threshold δ.

This works fine, but there are also some issues. First, why only look at the dips following peaks, and not the ones preceding a peak? Second, small fluctuations in intensity can create multiple peaks in a short time interval. They might all have small dips and thus not be selected, even though the highest has a big dip if you ignore the other ones. So that is exactly what I propose: to first drop peaks with too *small* a dip. That leads to the following algorithm. First, sweep through the peaks $p_1, \ldots, p_n$ and for every point check the drop $|p_i - v_i|$. If it is below some ϵ, remove the lowest of the peaks p_i and p_{i+1}. Then continue with the point that is left (you might have to consider the same point multiple times, but that's the idea). After this, you repeat the original algorithm with a small adaptation: you now check if $|p_i - v_i| > \delta_1$, but also if $|p_i - v_{i-1}| > \delta_{-1}$. In other words: both the following and the preceding dip must be deep enough.

2.1 Optimization and evaluation

To compare this with the original method, I first had to find optimal parameter settings for ϵ, δ_1 and δ_2. After manually annotating 99 English stimuli with approximate nucleus intervals (rather than points) I randomly divided the stimuli in training (70%) and test (30%). The parameter estimation was done using a naive grid search, with $\delta_{1,2} = 0.5, 1, 1.5, \ldots, 5$ and $\epsilon = 0, 0.5, \ldots, 10$.

1 In fact, that study lead me to believe that nuclei might a crucial part.

2 But note that state of the art syllable segmentation relies on essentially the same intensity and pitch data and it thus very similar in spirit.

As an error measure, I compared the identified nuclei with the annotated nucleus intervals. I counted the number of *duplicate nuclei* (multiple nuclei where there should be one), *superfluous nuclei* (nuclei where there should be none) and *missing nuclei*. I averaged those counts over all training stimuli and added the averages to obtain an error measure. It is intended to capture the average number of mistakes: missing or misplaced nuclei.

Interestingly, the error profile is very easy to interpret. [insert some graphs]. Minimum error 2.36.

This motivated a second, more refined grid search with $\epsilon = 3.5, 4.6, \ldots, 4.5$, $\delta_1 = 0, 0.1, 0.5$ and $\delta_2 = 1.5, 1.6, \ldots, 2.5$. The optimal settings were $\epsilon = 4$ and $\delta_2 = 1.8$ and δ_1 anything between $0.1, 0.2, \ldots 0.4$. In other words, δ_1 is irrelevant (as it is a minimum value, you can set it to 0). Interestingly, the minimum dip *before* the the peak (δ_2) is not irrelevant. This was not used in the original algorithm. However, the importance of the dip after the peak is captured by the larger value of ϵ.

Bibliography

Jong, Nivja H de and Ton Wempe (2009). "Praat script to detect syllable nuclei and measure speech rate automatically." In: *Behavior research methods* 41.2, pp. 385–90. DOI: 10.3758/BRM.41.2.385.

Mauch, Matthias et al. (2015). "Computer-aided Melody Note Transcription Using the Tony Software : Accuracy and Efficiency". In: *Proceedings of the First International Conference on Technologies for Music Notation and Representation*, p. 8.