Tony: a tool for melody transcription - Bug #875
the case of No Pitch Candidate

2014-02-24 07:09 PM - Matthias Mauch

Status:	Closed	**Start date:**	2014-02-24
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I have done some annotations now, and though I'm almost always happy with the editing functions I have, I'm sure that there'll come a time when the annotators will

- want to annotate a pitch
- pyin happens not to give them ANY! or presents them with one or several that are not in octave distance from the correct one

Presently, they'd be pretty much stuck then. Is there anything we'd want to do to prevent that?
I'm thinking we should. But how?

- scenario 1
  - select a region with no adequate pitch track
  - control-P (or whatever) will lead to the creation of a constant pitch track at 440 Hz
  - user moves individual frames
- scenario 2
  - anybody have a better idea than my crappy scenario 1?

History

#1 - 2014-02-24 08:21 PM - Rachel Bittner

Some ideas:

Scenario 2:

- select region
- control-P (to say "I am drawing")
- draw new pitch track by clicking and dragging
- hit return to confirm & delete any other pre-existing pitch track

Scenario 3:

- create new note using note editor
- press some shortcut key to draw a constant pitch track at the note's center frequency for the duration of the note
- hit return to confirm & delete any other pre-existing pitch track

Scenario 4:

- select region
- press some shortcut key to add new pitch
- prompt for frequency value (type or click?)
- draws constant pitch for duration of time range
- user manually adjusts at frame level?
- hit return to confirm & delete any other pre-existing pitch track

Thoughts?

#2 - 2014-02-24 08:28 PM - Justin Salamon

Hello all,

My inclination is to avoid as much as possible the scenario where the user draws something completely manually. These annotations are supposed to be used as a ground truth, and letting users draw their own pitch track can lead to some pretty inaccurate results.

My vision for scenario 2 would be where the user can select a bounding box (in time / fqr), and pYin is asked for a pitch sequence in there. Shouldn't it be possible to force the algorithm to return the highest scoring pitch value within the bounding box at each point in time, even if it's a poor score?

#3 - 2014-02-27 10:20 PM - Rachel Bittner

Just a quick note that I have found a couple of cases where there is actually quite a need for this. In particular, I've run into lots of cases where there is either no candidate returned by pyin or the candidates given are not just octave shifts of the real pitch.

This is mainly happening (not surprisingly) for audio files where there is some bleed from other instruments in the track.

#4 - 2014-02-27 10:27 PM - Justin Salamon

Hmmm I see your point.

Is the "right" pitch visible in the spectrogram view? We could have a draw option with 'snap to spectral peaks' for instance. Though I think that is probably more work than we can afford to do given that we want to get started on annotation next week.

j

#5 - 2014-02-28 10:46 AM - Matthias Mauch

Rachel Bittner wrote:

> *Just a quick note that I have found a couple of cases where there is actually quite a need for this. In particular, I've run into lots of cases where there is either no candidate returned by pyin or the candidates given are not just octave shifts of the real pitch.*
>
> *This is mainly happening (not surprisingly) for audio files where there is some bleed from other instruments in the track.*

Ok, Incidentally, I think the newest version should be better at that than the version I suspect you're using. I will have to chat with Chris, probably on what the "definitive" pyin version we should use is. (The one I'm currently using is the latest I pushed to the pyin repo, to the branch "tony").

#6 - 2014-02-28 11:11 AM - Matthias Mauch

> *My vision for scenario 2 would be where the user can select a bounding box (in time / fqr), and pYin is asked for a pitch sequence in there. Shouldn't it be possible to force the algorithm to return the highest scoring pitch value within the bounding box at each point in time, even if it's a poor score?*

Yes, that's possible. Options are:

- return best value per frame (no smoothing)
  - advantage: no weird hard-to-explain jumps introduced by smoothing over a restricted area.
  - disadvantage: pitch track might not be continuous

- return best smoothed pitch track over values in range
  - advantage: you'll have a mostly smooth pitch tracks
  - disadvantage: you might get "unexplainable" jumps if the actual pitch crosses your selection boundary (this is what we saw earlier with the jagged pyin candidates)

Matthias

#7 - 2014-02-28 06:09 PM - Matthias Mauch

> - *return best value per frame (no smoothing)*
>   - *advantage: no weird hard-to-explain jumps introduced by smoothing over a restricted area.*
>   - *disadvantage: pitch track might not be continuous*

I've just made a new YIN-derivative plugin that does just that:
http://code.soundsoftware.ac.uk/projects/pyin/repository/entry/YinVampFreqConstrained.cpp
Returns the frequency corresponding to the best value (according to the YIN difference function) in the frequency range provided.
So if we have a way of calling that from the interface, it might just work.

#8 - 2014-02-28 06:23 PM - Matthias Mauch

- Priority changed from Normal to High

Set this to high, since I believe it's the only missing piece in the puzzle to make a usable program, I think.

#9 - 2014-02-28 06:31 PM - Chris Cannam

Noted. I'll see if I can think of a way to deal with this on the Tony side.

#10 - 2014-03-04 11:05 AM - Chris Cannam

- Priority changed from High to Immediate

#11 - 2014-03-05 10:03 AM - Chris Cannam

- File Screenshot - 050314 - 09_50_12.png added
- Assignee set to Matthias Mauch

You can now use shift-mouse in the main pane to select a time-frequency region and Tony will calculate (and display immediately) pitch candidates in that region using the new yinfc plugin in the pyin library.

The interaction model perhaps isn't ideal yet, not least because the candidates appear straight away when you do this, rather than appearing only on ctrl+return as they do in normal selection.

However, more pressingly, the results you get simply don't appear to be very good. If I highlight a region that contains a visible harmonic in the spectrogram, I'm as likely to see a scattering of points above and below that harmonic as I am to see anything that looks like a clean pitch track. (See attached picture, choosing second harmonic from an arbitrary note in happy_birthday_mm_normal.wav)

So I'm assigning over to Matthias while I work on something else -- can you take a look and see what you think should be done? Note that the choice of plugin & processing parameters are made in Analyser::reAnalyseSelection.

#12 - 2014-03-05 10:55 AM - Chris Cannam

(Note that the constrained-pitch candidate no longer appears straight away, in the latest commit -- as a side-effect of work on #885, you now have to hit Ctrl+Return after selecting the region just as you do with a normal selection. The UI feedback could still be clearer as it shows only a time selection after you have finished dragging the rectangle, so that's still a work in progress, but the quality of the actual results remains more pressing.)

#13 - 2014-03-05 05:02 PM - Matthias Mauch

> *However, more pressingly, the results you get simply don't appear to be very good. If I highlight a region that contains a visible harmonic in the spectrogram, I'm as likely to see a scattering of points above and below that harmonic as I am to see anything that looks like a clean pitch track. (See attached picture, choosing second harmonic from an arbitrary note in happy_birthday_mm_normal.wav)*

I've played around a little, and in this case, I think this is more of a conceptional issue: YIN doesn't necessary have high salience at spectral peaks since it's a time-domain approach. Rather, it usually has peaks at sub-harmonics.

It does in fact produce nice results if you select a sub-harmonic. This is counter-intuitive only because we're displaying the spectrogram, which is not directly used in any of the computations. But I think this is not actually a problem because YIN/PYIN essentially never makes an octave error in the downward direction.

So I'd say this is not a bug, and I'd postpone the problem until someone finds a true pitch that isn't captured by the constrained pitch output.

#14 - 2014-03-05 05:12 PM - Matthias Mauch

- Priority changed from Immediate to Normal

#15 - 2014-03-07 10:25 AM - Chris Cannam

- File Screenshot - 070314 - 09_58_33.png added
- File caraclip.wav added

I see the logic for that case, but I still don't find the general situation very persuasive I'm afraid.

Have a look at the attached screenshot for another example. This is a singer accompanied by... um... something that sounds a bit like a harp[*]

You can see that the default pitch track (in black) has tracked the backing instrument rather than the vocal.

That's OK, because I can shift-click-drag to select a region containing only the vocal fundamental, stopping short of the backing instrument harmonics at both top and bottom... except it doesn't work, the result (in orange) still clings to the edges of the selected region as if it's still trying to follow the backing instrument even though it isn't really in range.

I can't for the life of me work out how to tell Tony what the actual vocal fundamental is in this example. Doesn't matter how tightly I box it in, the result always escapes and goes for the backing harmonic. The alternate candidates that I'm being offered in the normal select-ctrl+return workflow have the same problem, there are four of them and they all track different harmonics of the backing instrument.

Is this sort of (accompanied) music "in scope" for us?

(The piece is "Donald of Glencoe" by Cara Dillon - clip attached)

[*] it's very easy to find lyrics for songs online, there are dozens of competing listing sites, but I'd really like something that lists the instrumentation and personnel for a song. Without the sleevenotes I'm lost

#16 - 2014-03-07 10:53 AM - Chris Cannam

- File Screenshot - 070314 - 10_30_53.png added

- File use-simple-cepstrum.diff.txt added

For comparison, the attached screenshot shows the result of making Tony use the existing Vamp Simple Cepstrum plugin to carry out the constrained analysis. I also attach the diff to Tony that makes it do that if you want to try it yourself.

I don't think the Vamp Simple Cepstrum is the ideal method for this -- it's not reliable enough (it often misses entirely) and it lacks resolution (in the y axis). But I do think there's a good case to be made that if you're selecting a region graphically, on the screen, you expect the results that come back to correspond to something you are actually looking at and a simple harmonic-spectrum or cepstral method is more likely to do what you expect in that situation.

#17 - 2014-03-07 11:15 AM - Matthias Mauch

That's cool, I don't mind that. Doesn't have to be all pyin.

The only concern might be mixing algorithms. That might not matter much given that the user has fiddled with the stuff anyway, only that you will get different results if you apply it to the same pitch track.

I'm a bit unsure about patches, so I just trust it works ok for now. Happy for you to commit that.

#18 - 2014-03-07 11:22 AM - Chris Cannam

Hm, I wasn't really proposing a solution -- as I said, I don't think the simple cepstrum plugin is entirely up to it either.

I do think it's probably better than using pyin for this, though I wouldn't mind a second opinion.

Not sure whether we have a harmonic-spectrum plugin with frequency range limits ready to roll, though it'd be easy enough to make one. Maybe I should try that too.

Anyway I take your comment as saying you're busy with something else and would be OK with seeing this at least provisionally changed if it looked appropriate.

#19 - 2014-03-07 04:08 PM - Justin Salamon

Hello all,

Actually, the other day Rachel and I were looking at a real example of an instrumental melody where the estimated pitch was below the real fundamental, but not at an octave relationship, and none of the candidates matched it. Similar to what Chris was experiencing, selecting a bounding box around the true location of the pitch curve did not yield coherent pitch values either.

We were thinking that having alternative pitch tracking algorithms available, which may be robust to different types of errors, would probably be the best solution, but also one that's a little late to implement. In fact, this is something we've all discussed at at the early stages of our collaboration, but it seemed like pYin wasn't making any unrecoverable errors so we'd abandoned the idea.

If we're reviving this idea, I'd like to add my 2 cents: first, I think we probably want a frequency-domain algorithm (as a time-domain algorithm is likely to make the same mistakes as yin). Given this, I'd consider the SWIPE algorithm by Camacho: at least in his evaluations it outperforms a variety of pitch trackers (including the classic yin algorithm). The thesis includes a matlab implementation that we could port, I have also found a python implementation here, and perhaps more interestingly a C implementation here.

I'm still a little weary about this since we won't have time to evaluate any pitch algorithm we implement thoroughly (or at all), but if we limit its use to the places where pYin fails, it's probably OK?

#20 - 2014-03-07 04:22 PM - Chris Cannam

Hi all -- I've just pushed a change that makes Tony use the newly-written Constrained Harmonic Peak plugin for frequency-constrained analysis.

This plugin implements a simple harmonic product spectrum. It relies entirely on being given a reasonable bound for the frequency range (i.e. it is only suitable for, and is only used in, the case where the user draws a bounding box using shift-click-drag in Tony).

I agree that we shouldn't be trying difficult or untested algorithms at this point -- and I didn't really want to introduce an untested plugin, but I couldn't find one that was quite suitable in terms of the interface available. What I do think is that, if we're going to introduce a new method, it should be one so simple that we can review the code easily and test that it is actually right.

So please do take a look at the CHP code and tell me if you think it looks wrong! Or if there is any other equally simple method you'd prefer. Must be really simple though, one spectral input frame at a time and no state.

One problem with this is that I'm not sure about the whole business of making sure its outputs are aligned with pyin's correctly in time. The analyser is quite selective about its timing in order to work nicely with pyin, perhaps it won't work so well with this? This one just uses frequency-domain input (timestamp aligned with centre of frame).

As far as providing other more sophisticated pitch trackers goes -- I think it'd be nice to do that but we should hold off for now and consider providing them later as alternatives within the whole application.

#21 - 2014-03-07 04:24 PM - Chris Cannam

- Priority changed from Normal to Immediate

#22 - 2014-03-07 04:59 PM - Justin Salamon

This sounds like a sensible approach to me! Of course we should try the resulting plugin with a couple of real problematic recordings and see if it addresses the problem, but assuming it does, it makes sense to use this approach.

In fact, what you're suggesting is basically to compute a harmonic-summation based salience function in the range indicated by the user and to return the most salient pitch candidate within the range, which is not too far from what MELODIA would do for a very constrained pitch range :)

I had a look at the code, the core bit is this one:

```
    for (int j = 1; j <= m_harmonics; ++j) {
            if (j * bin > hs) break;
            hps[i] *= mags[j * bin];
            ++contributing;
        }
        if (hps[i] <= 0.0) {
            hps[i] = -120.0;
        } else {
            hps[i] = 20.0 / contributing * log10(hps[i]);
        }
```

If I understand correctly, you're summing the dB amplitudes of the harmonics, and then taking the average amplitude by dividing by the number of

harmonics used in the summation, and that's the value that you compare over the given pitch range to choose the max?

I think that makes sense. In MELODIA I use the linear peak magnitudes rather than dB, and instead of dividing the sum by the number of harmonics, I use a weighting function so that the higher the harmonic the less it contributes: 0.8^(nHarm -1) (in this formula the fundamental is nHarm=1). But my salience function is designed for polyphonic signal... in the quasi-monophonic case (melody + bleed) a simpler strategy should probably work. The only snag might be that you assume the harmonic will be located at an exact multiple of the candidate pitch, which it won't always be. Klapuri for instance handles this by defining a frequency range around the estimated location of each harmonic and taking the max bin value within this range - again, maybe not necessary, but potentially more robust to slightly inharmonic sounds.

To summarize all of my blurb, I think this is a good approach, lets test it on some files to see whether the proposed harmonic summation needs any tweaking or does the trick. Good stuff!

#23 - 2014-03-07 05:12 PM - Matthias Mauch

Cool stuff.

It pains me a bit that I've spent quite a bit of time on this YIN thing, only to realise more and more its failings! I guess I have to live and learn.

I've got another one, will make that one a new issue...

#24 - 2014-03-07 06:25 PM - Chris Cannam

- Assignee changed from Matthias Mauch to Justin Salamon

Justin -- yes, I think your summary is correct. Another way to put it is that I'm copying whatever Craig Sapp did in his Harmonic Spectrum Mazurka plugin. (when running in "product" processing mode) It seemed reasonable to me.

(The Mazurka plugin would almost fit here, but it wants its pitch range in MIDI units and its licence is ambiguous.)

I'm not sure how significant the harmonic content really is, given that we've been given a frequency range anyway. The interpolated peak within that range might be just as good.

It'd be great if you had a while to take a look at the results you get with it and (if necessary) mess with (or replace) the code to improve it. I can of course add you to the project if you have anything that works nicely that you'd like to push.

btw the plugin calls for a default block size of 2048 but Tony is actually running it at 4096 (though with the same hop as pyin). No problem if you want to change that either.

#25 - 2014-03-10 04:00 PM - Justin Salamon

I tried it quickly, it seems to be producing sensible results. I'll only be able to give proper feedback when we start annotation work.

One thing though - the timestamps - normally in SV a plugin that takes frequency domain input returns a timestamps that is at the center of the frame (which is what we want). Is this still the case here? In terms of window size, 4096 (assuming fs=44100) might be a little big - can we use 2048 instead? That'll be more consistent with pYin as well I think.

#26 - 2014-03-10 05:37 PM - Rachel Bittner

- File MichaelKropf_AllGoodThings_Premix_02.wav added

I like this solution a lot. I played with it on some of the really hard examples (like the one attached). The new solution makes it possible to annotate this (though it's a pain since it's really not monophonic).

As a side note, the data we end up using is going to be cleaned up so things won't be this far from monophonic. But I bring this example up as an extreme case, and even here Tony is usable.

#27 - 2014-03-12 10:22 AM - Chris Cannam

- Priority changed from Immediate to Normal

#28 - 2014-04-02 09:38 AM - Chris Cannam

- Status changed from New to Closed

- Assignee deleted (Justin Salamon)

Closing this one. Please open any remaining problems as separate issues.

Downloads

Screenshot - 050314 - 09_50_12.png	22.5 KB	2014-03-05	Chris Cannam
Screenshot - 070314 - 09_58_33.png	37.9 KB	2014-03-07	Chris Cannam
caraclip.wav	2.22 MB	2014-03-07	Chris Cannam
Screenshot - 070314 - 10_30_53.png	39.5 KB	2014-03-07	Chris Cannam
use-simple-cepstrum.diff.txt	915 Bytes	2014-03-07	Chris Cannam
MichaelKropf_AllGoodThings_Premix_02.wav	35.9 MB	2014-03-10	Rachel Bittner