Computer-aided Melody Note Transcription Using the Tony Software: Accuracy and Efficiency

Matthias Mauch,[1] **Chris Cannam,**[1] **Rachel Bittner,**[2] **George Fazekas,**[1]
Justin Salamon,[2] **Jiajie Dai,**[1] **Juan Bello,**[2] **Simon Dixon**[1]
[1)] Queen Mary University of London, [2)] New York University

ABSTRACT

We present Tony, a software tool for the interactive annotation of melodies from monophonic audio recordings, and evaluate its usability and the accuracy of its note extraction method. The scientific study of acoustic performances of melodies, whether sung or played, requires the accurate transcription of notes and pitches. To achieve the desired transcription accuracy for a particular application, researchers manually correct results obtained by automatic methods. Tony is an interactive tool directly aimed at making this correction task efficient. It provides (a) state-of-the art algorithms for pitch and note estimation, (b) visual and auditory feedback for easy error-spotting, (c) an intelligent graphical user interface through which the user can rapidly correct estimation errors, (d) extensive export functions enabling further processing in other applications. We show that Tony's built in automatic note transcription method compares favourably with existing tools. We report how long it takes to annotate recordings on a set of 96 solo vocal recordings and study the effect of piece, the number of edits made and the annotator's increasing mastery of the software. Tony is Open Source software, with source code and compiled binaries for Windows, Mac OS X and Linux available from https://code.soundsoftware.ac.uk/projects/tony/.

1. INTRODUCTION

Our goal is to make the scientific annotation of melodic content, and especially the estimation of note pitches in singing, more efficient. A number of well-known digital signal processing methods have been successfully applied to measuring singing pitch precisely and unambiguously, e.g. [1,2]. While their accuracy is sufficient for many applications, arriving at a satisfactory annotation often requires significant manual adjustment on the part of the researcher. This need for adjustment is even more pronounced when the aim is to transcribe discrete notes. Performing such adjustments take much time and effort, especially in the absence of a user-friendly interface.

The main contributions of this paper are (1) the presentation of the Tony user interface aimed at streamlining the melody annotation process, (2) the new note transcription algorithm it uses (implemented in the pYIN Vamp plugin), and (3) an evaluation of Tony's utility in terms of note transcription accuracy and the effort required for note annotation in a real-world use case. Features and design described in this paper reflect Tony version 1.0 except where noted.

software	URL
Tony	https://code.soundsoftware.ac.uk/projects/tony
pYIN	https://code.soundsoftware.ac.uk/projects/pyin
Pitch Estimator	https://code.soundsoftware.ac.uk/projects/chp
Sonic Visualiser Libraries	https://code.soundsoftware.ac.uk/projects/sv

Table 1: Software availability.

2. BACKGROUND

Music informatics researchers, music psychologists and anyone interested in the analysis of pitch and intonation routinely use software programs to annotate and transcribe melodies in audio recordings. The two main objects of interest are the pitch track, which traces the fundamental frequency (F0) contours of pitched sounds in smooth, continuous lines, and the note track, a sequence of discrete note events that roughly correspond to notes in a musical score. In order to find out which tools are used we conducted an online survey that was sent out through several channels including the *ISMIR Community*, *Auditory* and *music-dsp* mailing lists. [1]

We obtained 31 responses with a strong bias towards researchers in music informatics (see Table 2). Most of the participants were from academic institutions (27; 87%), of which students were the greatest contingent (11; 35%). Four participants were from industry (13%). Experience with pitch and note representations was nearly evenly distributed (58% and 52%, respectively, including those who had experience with both kinds of annotation).

We asked the participants which tools they are aware of. Responses included a large variety of tools, which we separated into user-interface-based software and signal pro-

[1] The survey questions are given in Appendix A.

Field of work		Position	
Music Inf./MIR	17 (55%)	Student	11 (35%)
Musicology	4 (13%)	Faculty Member	10 (32%)
Bioacoustics	3 (10%)	Post-doc	6 (19%)
Speech Processing	2 (5%)	Industry	4 (13%)
Experience			
Pitch track	18* (58%)		
Note track	16* (52%)		
Both	7 (23%)		
None	3 (10%)		

*) includes 7 who had experience with both pitch and note tracks.

Table 2: Participants of the survey. Top four responses for participant makeup.

cessing software without user interfaces (see Box 1).[2]

The tools with graphical user interfaces mentioned by survey participants were: Sonic Visualiser (12 participants), Praat (11), Custom-built (3), Melodyne (3), Raven (and Canary) (3), Tony (3), WaveSurfer (3), Cubase (2), and the following mentioned once: AudioSculpt, Adobe Audition, Audacity, Logic, Sound Analysis Pro, Tartini and Transcribe!.
The DSP algorithms mentioned by survey participants were: YIN (5 participants), Custom-built (3), Aubio (2), and all following ones mentioned once: AMPACT, AMT, DESAM Toolbox, MELODIA, MIR Toolbox, Tartini, TuneR, SampleSumo, silbido, STRAIGHT and SWIPE.

Box 1: Survey Results.

Our first observation is that despite the wide range of tools, there are some that were mentioned many more times than others: in the case of user interfaces these are Sonic Visualiser [3][3] and Praat [4][4], and in the case of DSP tools it is YIN [5]. None of the tools with user interfaces are specifically aimed at note and pitch transcription in music; some were originally aimed at the analysis of speech, e.g. Praat, others are generic music annotation tools, e.g. Sonic Visualiser and AudioSculpt [6]. In either case, the process of extracting note frequencies remains laborious and can take many times the duration of the recording. As a consequence, many researchers use a chain of multiple tools in custom setups in which some parts are automatic (e.g. using AMPACT alignment [7]), as we have previously done ourselves [8]. Commercial tools such as Melodyne,[5] Songs2See[6] and Sing&See[7] serve similar but incompatible purposes. Melodyne in particular offers a very sleek interface, but frequency estimation procedures are not public (proprietary code), notes cannot be sonified, and clear-text export of note and pitch track data is not provided.

In summary, the survey further corroborated the impression gained during our own experiments on note intonation: a tool for efficient annotation of melodies is not available, and the apparent interest in the scientific study of melody provides ample demand to create just such a tool. We therefore set out to create Tony, a tool that focusses on melodic annotation (as opposed to general audio annotation or polyphonic note annotation). The Tony tool is aimed at providing the following components: (a) state-of-the art algorithms for pitch and note estimation with high frequency resolution, (b) graphical user interface with visual and auditory feedback for easy error-spotting, (c) intelligent interactive interface for rapid correction of estimation errors, (d) extensive export functions enabling further processing in other applications. Lastly, the tool should be freely available to anyone in the research community, as it already is (see Table 1). This paper demonstrates that the remaining requirements have also been met.

Any modern tool for melody annotation from audio requires signal processing tools for pitch (or fundamental frequency, F0) estimation and note transcription. We are concerned here with estimation from monophonic audio, not with the estimation of the predominant melody from a polyphonic mixture (e.g. [9, 10]). Several solutions to the problem of F0 estimation have been proposed, including mechanical contraptions dating back as far as the early 20th century [11]. Recently, the area of speech processing has generated several methods that have considerably advanced the state of the art [4, 5, 12, 13]. Among these, the YIN fundamental frequency estimator [5] has gained popularity beyond the speech processing community, especially in the analysis of singing [14, 15] (also, see survey above). Babacan *et al.* [16] provide an overview of the performance of F0 trackers on singing, in which YIN is shown to be state of the art, and particularly effective at fine pitch recognition. More recently, our own pYIN pitch track estimator has been shown to be robust against several kinds of degradations [17] and to be one of the most accurate pitch transcribers, especially for query-by-singing applications [18] (alongside the MELODIA pitch tracker [10]).

The transcription of melodic *notes* has received far less attention than pitch tracking—perhaps because *polyphonic* note transcription [19, 20] was deemed the more exciting research problem—but several noteworthy methods exist [2, 21, 22]. We have implemented our own note transcription method intended for use in Tony, of which a previous version has been available as part of the pYIN Vamp plugin [17]. This is the first time pYIN note transcription has been presented and evaluated in a scientific paper.

3. METHOD

Tony implements several melody estimation methods: fully automatic pitch estimation and note tracking based on pYIN [17], and custom methods for interactive re-estimation. Tony resamples any input file to a rate of 44.1 kHz (if necessary), and the signal processing methods work on overlapping frames of 2048 samples (≈46 ms) with a hop size of 256 samples (≈6 ms).

[2] We are furthermore aware of tools for pitch track annotation [1] and pitch track and note annotation [2] that are not publicly available.

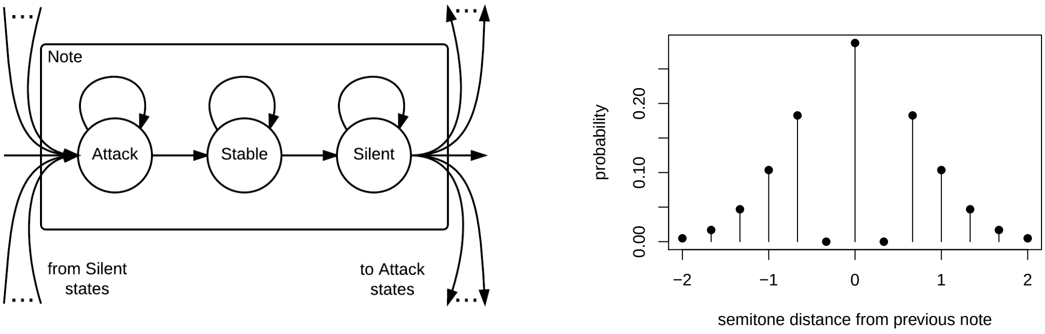
[3] http://www.sonicvisualiser.org/

[4] http://www.fon.hum.uva.nl/praat/

[5] http://www.celemony.com/

[6] http://www.songs2see.com/

[7] http://www.singandsee.com/



(a) Excerpt of the pYIN note transition network.

(b) Central part of the note transition probability function.

Figure 1: Transition model in pYIN note transcription module.

3.1 Pitch Estimation

We use the existing probabilistic YIN (pYIN) method [17] to extract a pitch track from monophonic audio recordings. The pYIN method is based on the YIN algorithm [5]. Conventional YIN has a single threshold parameter and produces a single pitch estimate. The first stage of pYIN calculates multiple pitch candidates with associated probabilities based on a distribution over many threshold parameter settings. In a second stage, these probabilities are used as observations in a hidden Markov model, which is then Viterbi-decoded to produce an improved pitch track. This pitch track is used in Tony, and is also the basis for the note detection algorithm described below.

3.2 Note Transcription

The note transcription method takes as an input the pYIN pitch track and outputs discrete notes on a continuous pitch scale, based on Viterbi-decoding of a second, independent hidden Markov model (HMM). Unlike other similar models, ours does not quantise the pitches to semitones, but instead allows a more fine-grained analysis. The HMM models pitches from MIDI pitch 35 (B1, ≈61 Hz) to MIDI pitch 85 (C♯6, ≈ 1109 Hz) at 3 steps per semitone, resulting in $n = 207$ distinct pitches. Following Ryynänen [21] we represent each pitch by three states representing attack, stable part and silence, respectively. The likelihood of a non-silent state emitting a pitch track frame with pitch q is modelled as a Gaussian distribution centered at the note's pitch p with a standard deviation of σ semitones, i.e.

$$P(n_p|q) = v \cdot \left(\frac{1}{z} \left[\phi_{p,\sigma}(q) \right]^{\tau} \right) \tag{1}$$

where n_p is a state modelling the MIDI pitch p, z is a normalising constant and the parameter $0 < \tau < 1$ controls how much the pitch estimate is trusted; we set $\tau = 0.1$. The probability of unvoiced states is set to $P(\text{unvoiced}|q) = (1 - v)/n$, i.e. they sum to their combined likelihood of $(1 - v)$ and $v = 0.5$ is the prior likelihood of a frame being voiced. The standard deviation σ varies depending on the state: attack states have a larger standard deviation ($\sigma = 5$ semitones) than stable parts ($\sigma = 0.9$). This models that the beginnings of notes and note transitions tend to vary more in pitch than the main, stable parts of notes.

The transition model imposes continuity and reasonable pitch transitions. Figure 1a shows a single note model, with connections to other notes. Within a note we use a 3-state left-to-right HMM consisting of Attack, Stable and Silent states. These states are characterised by high self-transition probability (0.9, 0.99 and 0.9999 for the three note states, respectively), to ensure continuity. Within a note, the only possibility other than self-transition is to progress to the next state. The last note state the Silent state, allows transitions to many different Attack states of other notes. Like the musicological model in Ryynänen and Klapuri's approach [21] we provide likelihoods for note transitions. Unlike their approach, we do not deal with notes quantised to the integer MIDI scale, and so we decided to go for a simpler heuristic that would only take into account three factors: (1) a note's pitch has to be either the same as the preceding note or at least 2/3 semitones different; (2) small pitch changes are more likely than larger ones; (3) the maximum pitch difference between two consecutive notes is 13 semitones. A part of the transition distribution to notes with nearby pitches is illustrated in Figure 1b.

3.3 Note Post-processing

We employ two post-processing steps. The first, amplitude-based onset segmentation helps separate consecutive notes (syllables) of similar pitches as follows. We calculate the root mean square (RMS, i.e. average) amplitude denoted by a_i in every frame i. In order to estimate the amplitude rise around a particular frame i we calculate the ratio of the RMS values between the frames either side

$$r = \frac{a_{i+1}}{a_{i-1}} \tag{2}$$

Given a sensitivity parameter s, any rise with $1/r < s$ is considered part of an onset,[8] and the frame $i - 2$ is set to unvoiced, thus creating a gap within any existing note.

[8] The inverse $1/r$ is used in order for s to correspond to sensitivity.

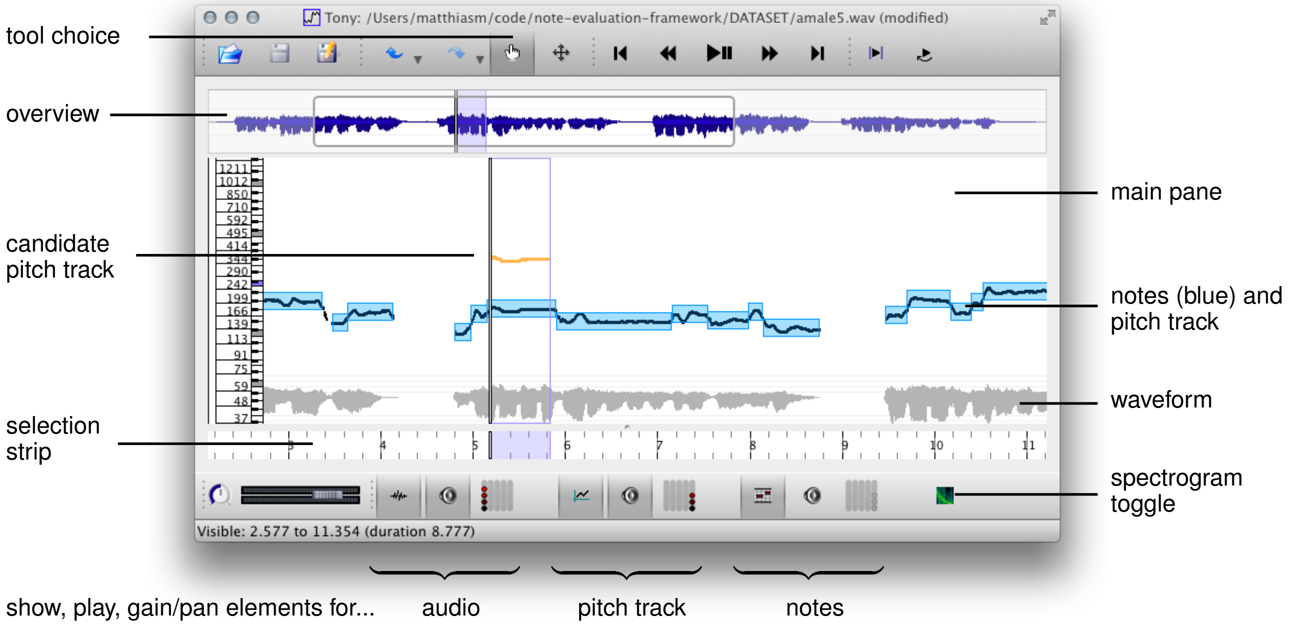

Tony: /Users/matthiasm/code/note-evaluation-framework/DATASET/amale5.wav (modified)
tool choice
overview
candidate pitch track
selection strip
main pane
notes (blue) and pitch track
waveform
spectrogram toggle
1211
1012
850
710
592
495
414
344
290
242
199
166
139
113
91
75
59
48
37
3
4
5
6
7
8
9
10
11
Visible: 2.577 to 11.354 (duration 8.777)
show, play, gain/pan elements for...
audio
pitch track
notes

Figure 2: Graphical User Interface with key elements highlighted.

If no note is present, nothing changes, i.e. no additional notes are introduced in this onset detection step. The second post-processing step, minimum duration pruning, simply discards notes shorter than a threshold, usually chosen around 100 ms.

3.4 Semi-automatic Pitch Track Re-estimation

In addition to fully manual editing of notes (Section 3.4.2), the user can also change the pitch track. However, since human beings do not directly perceive pitch tracks, Tony offers pitch track candidates which users can choose from. Two methods are available: multiple alternative pYIN pitch tracks on a user-selected time interval, and a single pitch track on a user-selected time-pitch rectangle.

3.4.1 Multiple pYIN pitch tracks

In order to extract multiple pitch tracks, the pYIN method is modified such that its second stage runs multiple times with different frequency ranges emphasised. The intended use of this is to correct pitches over short time intervals. As in the default version, the first pYIN stage extracts multiple pitch candidates m_i (given in floating point MIDI pitches) for every frame, with associated probabilities p_i. Depending on the frequency range, these candidate probabilties are now weighted by a Gaussian distribution centered at $c_j = 48 + 3 \times j$, $j = 1, \ldots, 13$, for the j^{th} frequency range, i.e. the new candidate pitch probabilities are

$$p_{ij} = p_i \times \phi_{c_j,\sigma_r}(m_i), \tag{3}$$

where $\phi(\cdot)$ is the Gaussian probability density function and $\sigma_r = 8$ is the pitch standard deviation, indicating the frequency width of the range. With these modified pitch probabilities, the Viterbi decoding is carried out as usual, leading to a total of 13 pitch tracks.

Finally, duplicate pitch tracks among those from the 13 ranges are eliminated. Two pitch tracks are classified as duplicates if at least 80% of their pitches coincide. Among each duplicate pair, the pitch track with the shorter time coverage is eliminated.

3.4.2 Pitch track in time-pitch rectangle

In some cases, the desired pitch track is not among those offered by the method described in Section 3.4.1. In such cases we use a YIN-independent method of finding pitches based on a simple harmonic product spectrum [23]. When using this method, the user provides the pitch and time range (a rectangle), and for every frame the method returns the pitch with the maximum harmonic product spectral value (or no pitch, if the maximum occurs at the upper or lower boundary of the pitch range). This way even subtle pitches can be annotated provided that they are local maxima of the harmonic product spectrum.

4. USER INTERFACE

Figure 2 is a screenshot of the Tony user interface. The basic interface components as well as the underlying audio engine and other core components are well tested as they come from the mature code base of Sonic Visualiser (see also Table 1). Tony differs from the other tools in that it is designed for musical note sequences, not general pitch events, and intentionally restricted to the annotation of single melodies. This specialisation has informed many of our design choices. Below we highlight several key aspects of the Tony interface.

4.1 Graphical Interface

While graphical interface components from Sonic Visualiser have been re-used, the focus on a single task has al-

	Group.1	Overall. Acc.	Raw. Pitch. Acc.	Vo. False Alarm	Vo. Recall	F COnPOff	F COnP	F COn
1	melotranscript	0.80	0.87	0.37	0.97	0.45	0.57	0.63
2	ryynanen	0.72	0.76	0.37	0.94	0.30	0.47	0.64
3	smstools	0.80	0.88	0.41	**0.99**	0.39	0.55	0.66
4	pYIN s=0.0, prn=0.00	0.83	**0.91**	0.37	0.98	0.38	0.56	0.61
5	pYIN s=0.0, prn=0.07	0.84	**0.91**	0.34	0.98	0.40	0.59	0.64
6	pYIN s=0.0, prn=0.10	0.84	**0.91**	0.33	0.97	0.41	0.60	0.64
7	pYIN s=0.0, prn=0.15	0.84	0.90	0.32	0.96	0.41	0.60	0.63
8	pYIN s=0.6, prn=0.00	0.84	**0.91**	0.35	0.98	0.38	0.56	0.61
9	pYIN s=0.6, prn=0.07	0.84	**0.91**	0.32	0.97	0.43	0.62	0.67
10	pYIN s=0.6, prn=0.10	**0.85**	**0.91**	0.31	0.97	0.44	0.62	0.67
11	pYIN s=0.6, prn=0.15	**0.85**	0.90	0.29	0.95	0.44	0.62	0.65
12	pYIN s=0.7, prn=0.00	0.83	0.90	0.33	0.97	0.39	0.54	0.61
13	pYIN s=0.7, prn=0.07	**0.85**	**0.91**	0.30	0.97	0.46	0.63	0.69
14	pYIN s=0.7, prn=0.10	**0.85**	0.90	0.29	0.96	0.47	0.64	0.69
15	pYIN s=0.7, prn=0.15	**0.85**	0.89	0.27	0.94	0.47	0.64	0.67
16	pYIN s=0.8, prn=0.00	0.84	0.89	0.28	0.96	0.39	0.52	0.61
17	pYIN s=0.8, prn=0.07	**0.85**	0.89	0.25	0.95	0.48	0.66	**0.73**
18	pYIN s=0.8, prn=0.10	**0.85**	0.89	0.24	0.94	0.49	**0.68**	**0.73**
19	pYIN s=0.8, prn=0.15	**0.85**	0.87	**0.22**	0.91	**0.50**	0.67	0.71

Table 3: Results for fully-automatic melody note transcription.

lowed us to combine all relevant visualisation components into a single pane: pitch track, note track, spectrogram and the waveform. Visibilty of all can be toggled. The focus on single melodies meant that we could design a special note layer with non-overlapping notes. This averts possible annotation errors from overlapping pitches.

As soon as the user opens an audio file, melodic representations of pitch track and notes are calculated using the methods described in Sections 3.1 and 3.2. This contrasts with general tools like Praat, Sonic Visualiser or AudioSculpt, which offer a range of processing options the user has to select from. This is avoided in Tony, since the analysis objective is known in advance. However, the user has some control over the analysis parameters via the menu and can re-run the analysis with the parameters changed.

Editing pitch tracks and notes is organised separately. Note edits concern only the placement and duration of notes in time, and their pitch is calculated on the fly as the median of the underlying pitch track. Any corrections in the pitch dimension are carried out via the pitch track.

In order to select pitches or notes the user selects a time interval, either via the Selection Strip or via keyboard commands. Both pitch track and note track can then be manipulated based on the selection. The most simple pitch track actions are: choose higher/lower pitch (by octave) in the selected area; remove pitches in the selected area. For more sophisticated pitch correction, the user can request alternative pitch tracks in a selected time interval (see Section 3.4.1), or the single most likely pitch track in a time-pitch rectangle (see Section 3.4.2). Note actions are: Split, Merge, Delete, Create (including "form note from selection"), and Move (boundary). The note pitch is always the median of the pitch track estimates it covers and is updated in real-time.

4.2 Sound Interface

Tony provides auditory feedback by playing back the extracted pitch track as well as the note track alongside the original audio. Like the visual pitch track and note representations, playback (including that of the original recording) can be toggled using dedicated buttons in a toolbar (see Figure 2), giving users the choice to listen to any combination of representations they wish.

Sonification of the notes is realised as a wave table playback of an electric piano sound. The sound was especially synthesised for its neutral timbre and uniform evolution. Unlike other programs, synthesis in Tony is not constrained to integer MIDI notes, and can sonify subtle pitch differences as often occur in real-world performances. The pitch track is synthesised on the fly, using a sinusoidal additive synthesis of the first three harmonic partials.

5. EVALUATION

To assess the utility of Tony as a note transcription system, we conducted two experiments. First, we compared the underlying note transcription method to existing methods, using a publicly available dataset [24]. Second, in a real-world task an expert annotated notes for an intonation study using the Tony software, and we measured the time taken and the number of notes manipulated. The experimental results are given below.

5.1 Accuracy of Automatic Transcription

We used a test set of 38 pieces of solo vocal music (11 adult females, 13 adult males and 14 children) as collected and annotated in a previous study [24]. All files are sampled at 44.1 kHz. We also obtained note transcription results extracted by three other methods: Melotranscript [22], Gómez and Bonada [2], Ryynänen [21]. We ran 16 different versions of Tony's note transcription algorithm, a grid search of 4 parameter settings for each of the two post-processing methods. Minimum duration pruning was parametrised to 0 ms (no pruning), 70 ms, 100 ms and 150 ms. The amplitude-based onset segmentation parameter was varied as $s = 0,\ 0.6,\ 0.7$ and 0.8.

For frame-wise evaluation we used metrics from the evaluation of pitch tracks [25] as implemented in `mir_eval` [26], but applied them to notes by assigning to every frame the pitch of the note it is covered by. The results are listed in Table 3. The pYIN note transcriptions reach very high overall accuracy rates (0.83–0.85) throughout. The highest score of the other methods tested is 0.80.[9] Among the

[9] Note that Ryynanen's method outputs only integer MIDI notes, so

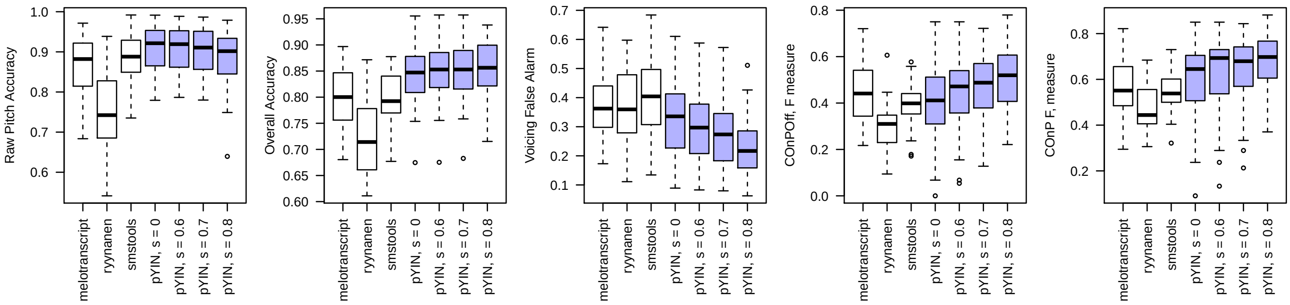


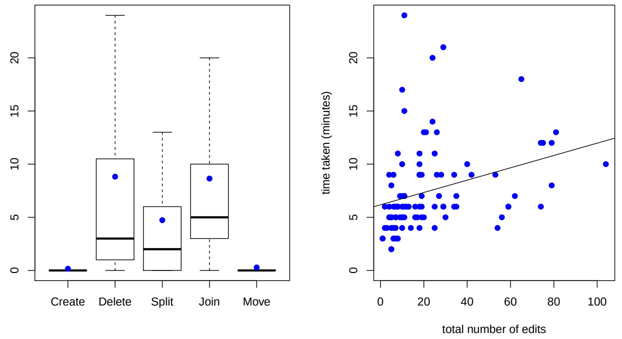
Figure 3: Results of existing algorithms and pYIN note transcription with minimum duration pruning at 0.1 s, showing, from left to right, raw pitch accuracy, overall accuracy, voicing false alarm, COnPOff F measure and COnP F measure.

pYIN versions tested, the best outcome was achieved by combining pruning of at least 100 ms and an onset sensitivity parameter of at least $s = 0.6$. The efficacy of the system results from high raw pitch accuracy (correct when there *is* a pitch), and low rate of voicing false alarm. There is, however, a tradeoff between the two: better raw pitch accuracy is achieved with low values of s, and lower false alarm rates with higher values of s. The algorithm smstools achieves perfect voicing recall at the price of having the highest voicing false alarm rate.

The results for note-based evaluation expose more subtle differences. The metric "COnPOff" [24], which takes into account correct note onset time (±5 ms), pitch (±0.5 semitones) and offset (± 20% of ground truth note duration), is the most demanding metric; "COnP" (correct onset and pitch) and "COn" (correct onset) are relaxed metrics. Here, we report F measures only. We observe that—without post-processing—the pYIN note transcription achieves values slightly worse than the best-performing algorithm (melotranscript). Considering the post-processed versions of pYIN, minimum duration pruning alone does not lead to substantial improvements. However, a combination of onset detection and minimum duration pruning leads to COnPOff F values of up to 0.50, compared to 0.38 for the baseline pYIN and 0.45 for the best other algorithm (melotranscript). This carries through to the more relaxed evaluation measures, where F values of the post-processed versions with at least 0.10 seconds pruning are always higher than the baseline pYIN algorithm and the other algorithms tested. Figure 3 shows all 100 ms-pruned pYIN results against other algorithms.

5.2 Effort of Manual Note Correction

In order to examine the usability of Tony we measured how editing affects the time taken to annotate tunes. We used recordings of amateur singing created for a different project, and one of us (JD) annotated them such that each final note annotation corresponded exactly to one ground truth note in the musical score matching her perception of the notes the singer was actually performing. The dataset consists of 96 recordings, with 32 singers performing three tunes from the musical *The Sound of Music*. The annotation was performed with an earlier version of Tony (0.6).



(a) Number of edits per recording. Box plots with mean values (dots).

(b) Time taken against edits.

Figure 4: Edit operations.

Tony offers five basic editing operations: Create, Delete, Split, Join, and Move (either left or right note boundary). We estimated the number of edits required, considering only timing adjustments (i.e. ignoring any changes to the pitch of a note).[10] The estimate is a custom edit distance implementation. First, we jointly represent the actual state of the note track (after automatic extraction) and the desired state of the note track as a string of tokens. Secondly, we define transformation rules that correspond to the five possible edit operations. The estimate of the number of edits performed by the user is then an automated calculation of a series of reductions to the source string in order to arrive at the target. In particular, if pYIN happened to perform a completely correct segmentation "out of the box", the edit count would be zero.

Figure 4a illustrates the distributions of edit counts in a box plot with added indicators of the mean. First of all, we notice that very few notes had to be Created (mean of 0.17 per recording) or Moved (0.28), and that Join (8.64) and Delete (8.82) are by far the most frequent edit operations, followed by Splits (4.73). As expected, the total number of edits correlates with the time taken to annotate the recordings (see Figure 4b).

Which other factors influence the annotation time taken? We use multivariate linear regression on the number of Creates, Deletes, Splits, Joins, Moves and familiarity with the Tony software (covariates), predicting the annotation

for the fine-grained analysis required here it may be at a disadvantage.

[10] At the time of the experiment we were not able to record the actual actions taken.

	Est. (seconds)	Std. Error	p value
(Intercept)	437.20	51.87	<0.01
Creates	145.08	42.77	<0.01
Deletes	3.51	1.82	0.06
Splits	5.58	2.95	0.06
Joins	3.18	2.35	0.18
Move	45.51	39.61	0.25
Familiarity	-2.31	0.82	0.01

Table 4: Effects on annotation time taken.

time (response). As expected, the results in Table 4 show that any type of editing increases annotation time, and that familiarity reduces annotation time. The baseline annotation time is 437 seconds, more than 7 minutes. (The mean duration of the pieces is 179 seconds, just under 3 minutes.) The result on Familiarity suggests that every day spent working with Tony reduces the time needed for annotation by 2.3 seconds. [11] The time taken for every Create action is 145 seconds, a huge amount of time, which can only be explained by the fact that this operation was very rare and only used on tracks that were very difficult anyway. Similar reasoning applies to the (boundary) Move operations, though the p value suggests that the estimate cannot be made with much confidence. The distinction between the remaining three edit operations is more helpful: each Delete and Join accounts for 3.5 seconds time added, but splits take much longer: 5.7 seconds. This is likely to result from the fact that the user has to position the play head or mouse pointer precisely at the split position, whereas joins and deletes require far less precise mouse actions. As Table 4 shows, most of the effects are at least moderately significant ($p < 0.1$), with the exception of number of Joins. The variance explained is $R^2 = 25\%$.

6. DISCUSSION

The results of the second experiment may well have impact on the design of future automatic melody transcription systems. They confirm the intuition that some edit actions take substantially more time for a human annotator to execute. For example, the fact that Merges are much cheaper than Splits suggests that high onset recall is more important than high onset precision.

We would also like to mention that we are aware that the accuracy of automatic transcription heavily depends on the material. The tools we evaluated (including existing algorithms) were well-suited for the database of singing we used; in other annotation experiments [27] it has become obvious that some instruments are more difficult to pitch-track. Furthermore, it is useful to bear in mind that the dataset we used is predominantly voiced, so the voicing false alarm outcomes may change on different data.

As evident from our survey (Box 1), early versions of Tony have already been used by the community. This includes our own use to create the MedleyDB resource [27], and some as yet unpublished internal singing intonation and violin vibrato experiments.

7. CONCLUSIONS

In this paper we have presented our new melody annotation software Tony, and its evaluation with respect to two aspects: firstly, an evaluation of the built-in note transcription system, and secondly a study on how manual edits and familiarity with the software influence annotation time.

The note transcription results suggest that the pYIN note transcription method employed in Tony is state-of-the-art, in terms of frame-wise accuracy and note-based evaluation. The study of manual edits shows the relative effort involved in different actions, revealing that Splits and Creates are particularly expensive edits. This suggests that for the task of note annotation, transcription systems should focus on voicing recall and note onset/offset accuracy.

In summary, we have presented a state-of-the-art note annotation system that provides researchers interested in melody with an efficient way of annotating their recordings. We hope that in the long run, this will create a surge in research and hence understanding of melody and intonation, especially in singing.

8. ACKNOWLEDGEMENTS

Thanks to Emilio Molina for kindly sharing the results of the experiments in his recent evaluation paper [24], and to all survey participants. Matthias Mauch is supported by a Royal Academy of Engineering Research Fellowship.

A. SURVEY QUESTIONS

- Have you ever used software to annotate pitch in audio recordings? (multiple choice)
- What software tools/solutions for pitch annotation exist? List tools that you are aware of. (free text)
- What characteristics of the tools would need to be improved to better suit your use case? (free text)
- Comments (free text)
- Your field of work (multiple choice)

B. REFERENCES

[1] S. Pant, V. Rao, and P. Rao, “A melody detection user interface for polyphonic music,” in *National Conference on Communications (NCC 2010)*, 2010, pp. 1–5.

[2] E. Gómez and J. Bonada, “Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing,” *Computer Music Journal*, vol. 37, no. 2, pp. 73–90, 2013.

[3] C. Cannam, C. Landone, M. B. Sandler, and J. P. Bello, “The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals.” in *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006, pp. 324–327.

[4] P. Boersma, “Praat, a system for doing phonetics by computer,” *Glot International*, vol. 5, no. 9/10, pp. 341–345, 2001.

[11] This is clearly only true within a finite study, since the reduction cannot continue forever. Annotations happened on 14 different days.

[5] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[6] N. Bogaards, A. Röbel, and X. Rodet, "Sound analysis and processing with AudioSculpt 2," in *Proceedings of the International Computer Music Conference (ICMC 2004)*, 2004.

[7] J. Devaney, M. Mandel, and I. Fujinaga, "A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT)," in *13th International Society of Music Information Retrieval Conference*, 2012, pp. 511–516.

[8] M. Mauch, K. Frieler, and S. Dixon, "Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory," *Journal of the Acoustical Society of America*, vol. 136, no. 1, pp. 401–411, 2014.

[9] M. Goto, "A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, vol. 43, no. 4, pp. 311–329, 2004.

[10] J. Salamon and E. Gòmez, "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2010.

[11] C. E. Seashore, "The tonoscope," *The Psychological Monographs*, vol. 16, no. 3, pp. 1–12, 1914.

[12] D. Talkin, "A robust algorithm for pitch tracking," in *Speech Coding and Synthesis*, 1995, pp. 495–518.

[13] H. Kawahara, J. Estill, and O. Fujimura, "Aperiodicity extraction and control using mixed mode excitation and group delay manipulation for a high quality speech analysis, modification and synthesis system STRAIGHT," *Proceedings of MAVEBA*, pp. 59–64, 2001.

[14] J. Devaney and D. Ellis, "Improving MIDI-audio alignment with audio features," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009, pp. 18–21.

[15] M. P. Ryynänen, "Probabilistic modelling of note events in the transcription of monophonic melodies," Master's thesis, Tampere University of Technology, 2004.

[16] O. Babacan, T. Drugman, N. D'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, 2013, pp. 7815–7819.

[17] M. Mauch and S. Dixon, "pYIN: a fundamental frequency estimator using probabilistic threshold distributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 2014, pp. 659–663.

[18] E. Molina, L. J. Tardón, I. Barbancho, and A. M. Barbancho, "The importance of F0 tracking in query-by-singing-humming," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 277–282.

[19] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[20] T. Cheng, S. Dixon, and M. Mauch, "A deterministic annealing EM algorithm for automatic music transcription," in *14th International Conference of the Society of Music Information Retrieval (ISMIR 2013)*, 2013, pp. 475–480.

[21] M. P. Ryynänen and A. P. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.

[22] T. De Mulder, J. Martens, M. Lesaffre, M. Leman, B. De Baets, and H. De Meyer, "Recent improvements of an auditory model based front-end for the transcription of vocal queries," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, vol. 4, 2004, pp. iv–257–iv–260.

[23] M. R. Schroeder, "Period histogram and product spectrum: New methods for fundamental-frequency measurement," *The Journal of the Acoustical Society of America*, vol. 43, no. 4, pp. 829–834, 1968.

[24] E. Molina, A. M. Barbancho, L. J. Tardón, and I. Barbancho, "Evaluation framework for automatic singing transcription," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 567–572.

[25] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[26] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "`mir_eval`: A transparent implementation of common MIR metrics," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 367–372.

[27] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDB: a multitrack dataset for annotation-intensive MIR research," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 155–160.