User-Specific Learning for Recognizing a Singer's Intended Pitch

Andrew Guillory
University of Washington
Seattle, WA
guillory@cs.washington.edu

Sumit Basu
Microsoft Research
Redmond, WA
sumitb@microsoft.com

Dan Morris
Microsoft Research
Redmond, WA
dan@microsoft.com

Abstract

We consider the problem of automatic vocal melody transcription: translating an audio recording of a sung melody into a musical score. While previous work has focused on finding the closest notes to the singer's tracked pitch, we instead seek to recover the melody the singer *intended* to sing. Often, the melody a singer intended to sing differs from what they actually sang; our hypothesis is that this occurs in a singer-specific way. For example, a given singer may often be flat in certain parts of her range, or another may have difficulty with certain intervals. We thus pursue methods for singer-specific training which use learning to combine different methods for pitch prediction. In our experiments with human subjects, we show that via a short training procedure we can learn a singer-specific pitch predictor and significantly improve transcription of intended pitch over other methods. For an average user, our method gives a 20 to 30 percent reduction in pitch classification errors with respect to a baseline method which is comparable to commercial voice transcription tools. For some users, we achieve even more dramatic reductions. Our best results come from a combination of singer-specific-learning with non-singer-specific feature selection. We also discuss the implications of our work for training more general control signals. We make our experimental data available to allow others to replicate or extend our results.

Introduction

Computer-based symbolic representations of music, such as MIDI (Musical Instrument Digital Interface, the most common standard for transmitting and storing symbolic music information), have been powerful tools in the creation of music for several decades. Musicians able to enter symbolic music with a musical instrument or a score-editing system have leveraged powerful synthesis and signal processing tools to create compelling audio output. However, this approach requires either advanced skill with an instrument or tedious manual score entry; both of these requirements may limit the creative expression and fluidity of music creation.

In order to make symbolic music processing tools more accessible and to allow more creativity and fluidity in symbolic music entry, existing work has attempted to replace the "musical instrument" in this process with a human voice by transcribing sung pitches into a symbolic melody. However, no system to date has been sufficiently accurate to replace a musical instrument as an entry system for symbolic music. The primary limitation has not been the determination of an audio stream's fundamental frequency, but rather the transformation of that frequency stream into a series of intended pitches and audio events.

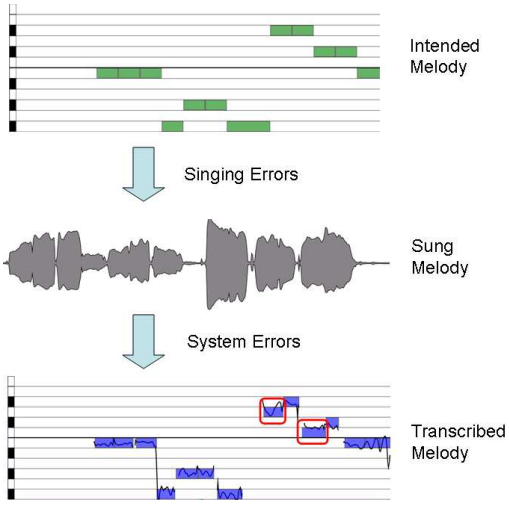

Figure 1: Automatic voice transcription process. We focus on singing errors as distinct from system errors.

Figure 1 shows an example of the voice transcription process. In this example, an attempt to transcribe the nursery rhyme "Old MacDonald Had a Farm", there are both system errors and singing errors in the final transcription. We call a transcription error a *system error* if it is the result of the system incorrectly recovering the actual sung melody. In our example, the final note of the melody is oversegmented by the note segmentation method, resulting in two extra notes. This is a system error, since inspection of the recording confirms the user did only sing one note. The second class of errors, *singing errors*, includes transcription errors due to differences between the sung melody and the intended melody. For example, a common singing error is to sing a note sharp or flat, resulting in a sung pitch one or two semitones off from the intended pitch. In our example, the singer is flat by

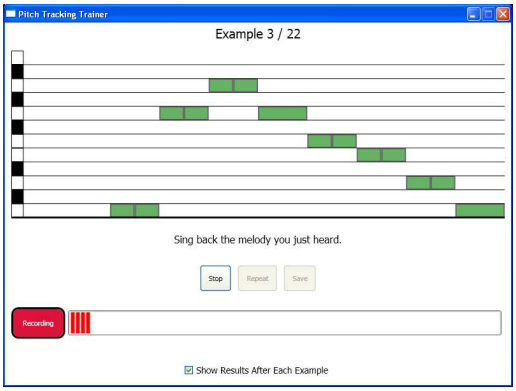

Figure 2: Screenshot of the data collection interface.

more than half a semitone on two notes (highlighted in red).

We hypothesize that a major limitation of previous research is the lack of a user-specific model accounting for singing errors. While previous work has assumed that the ideal transcription system would use canonical models that work for all users with no per-user training, we instead require a user to sing a short series of training melodies where the intended pitch and rhythm are known a priori, just as speech and handwriting recognition systems currently require user-specific training. By introducing this fundamentally new interaction paradigm, we hope to overcome the limitations that have until now prevented voice from being an effective melody input tool.

In this paper, we describe our training paradigm and our model for melodic transcription. We show that our system produces more accurate transcriptions than previous approaches. This work has implications for both music creation and the related field of music information retrieval, where recent work has attempted to search a melodic database for songs based on vocal input. We also discuss how related scenarios could benefit from similar approaches.

Data Collection Procedure

To learn a singer-specific mapping from a sung pitch to an intended pitch, we need a method for collecting recordings with known intended pitch. We have designed and implemented a procedure for quickly and easily collecting this training data without the need for hand annotation. The program asks the user to listen to a series of short (2 to 4 measure) melodies and then sing them back. At the start of the training procedure the program displays instructions to the user. Users are instructed to sing "doo" for each note. The program also asks users indicate their gender. Female participants are given examples with a higher pitch range. The examples we use are included in the online supplementary material.

At the start of each example, the program plays a synthesized version of the melody starting with the sound of the root note (the first note of the key the melody is in) played as a piano sound and a spoken countdown ("one, two, three, four"). The melody is then played as a synthesized horn sound along with a simple drum beat. While they listen to the melody, users are shown a visual representation of the melody in a piano roll format. Immediately after the melody finishes playing, on-screen instructions tell the user to sing back the melody. At the start of the recording phase of each example, the user hears the same root note and countdown. The user then sings along with the same drum beat, but does not hear the synthesized horn sound while singing. Figure 2 shows a screenshot of the main interface while recording. The online supplementary material also includes a video demonstrating the interface.

Recording stops automatically after the correct number of measures. When recording has finished, the user has the option to repeat the current example or save the recording and move on to the next example. Optionally, we give the user feedback about their singing accuracy after each example. This feedback can be skipped and is meant to engage the user and make the training procedure more enjoyable. Feedback is only displayed after the user has finished an example and chosen to move on in order to prevent the feedback from influencing whether or not a user repeats the example.

We have designed the training procedure in an attempt to reduce system errors and allow us to focus on singing errors and in particular pitch errors. We ask users to sing "doo" for each note so that the hard consonant sound makes it easy to identify the start of notes. By playing the backing drum track as users sing, we ensure that the recorded melodies are aligned with the melodies we ask them to sing. We minimized pitch perception issues by empirically comparing several synthesized instruments for pitch clarity; our experiments use a synthesized horn sound. Finally, by using short melodies, recording immediately after playback, and playing the melody's root note at the start of recording, we try to minimize the chance that the user forgets the melody or the key of the melody.

Non-Learning Methods for Pitch Prediction

Rather than building a learner from scratch, it seemed sensible to leverage the wide variety of existing non-learning-based methods for estimating pitch. We also expect that future practitioners will have new non-learning-based methods they will wish to try. We thus developed a formulation in which we could supply a large bag of candidate pitch predictions and let the algorithm sort out which will work best for the task in a singer-specific way.

We assume as input to our method a relatively accurate transcription of the raw frequencies present in a recording. We also assume this sequence is broken into segments corresponding to sung notes (see "Baseline Transcription Method"). Let p_i, then, refer to the pitch segment for the ith note in a melody and assume these pitch values are sampled at a regular interval (100Hz) and are represented on a log Hz scale (MIDI note numbers are integer values on this scale). Our goal is then to translate $p_1, p_2, ...$ into an estimate of the intended melody by labeling each note with an intended pitch estimate y_i. We assume the intended melody is a sequence of discrete pitch values, so y_i will ultimately be rounded to an integer pitch number on the chromatic scale (i.e. MIDI note number). Without loss of generalization,

all the methods we consider first predict y_i as a real value before rounding.

The remainder of this section will describe different non-learning methods for performing this labeling which will ultimately be used as features by our learner.

Simple Non-Learning-Based Methods

Perhaps the simplest method for estimating y_i from p_i is to simply take the median pitch value in p_i, which we write as $\text{median}(p_i)$. This heuristic assumes that the note contour for the sung pitch is roughly centered around the intended pitch for that note. We found $\text{median}(p_i)$ to be a relatively good estimate of intended pitch for singers with accurate pitch. The more complicated predictors we consider all use $\text{median}(p_i)$ as a proxy for sung pitch in forming more accurate estimates of intended pitch. Other predictors available to our learner take the mean, the maximum, or the minimum pitch value in p_i or compute the median of a small portion of p_i (for example the middle third of p_i).

More advanced predictors available to our learner use information from surrounding notes to adjust the pitch of a note. For example, if the sung pitch of the previous note is flat relative to the integer chromatic scale, we would expect the current note to also be flat by roughly the same amount. This intuition leads to the heuristic

$$y_i = \text{median}(p_i) + (\text{round}(\text{median}(p_{i-1})) - \text{median}(p_{i-1}))$$

We could also use a similar heuristic with the next note, p_{i+1}. Another similar heuristic assumes singers always intend to sing integer intervals and and rounds the difference between the previous and current sung pitches

$$y_i = \text{median}(p_{i-1}) + \text{round}(\text{median}(p_i) - \text{median}(p_{i-1}))$$

We can also apply a single shift to all pitch values in a sung melody. That is, predict $y_i = \text{median}(p_i) + \delta$ for some δ which is the same for all notes in a melody. Heuristics of this kind make sense if we expect that a singer's sung pitch differs from intended pitch by a constant amount throughout a sung melody. Among the choices for δ, we can assume notes in the melody are all roughly correct relative to the first note, giving $\delta = \text{round}(\text{median}(p_0)) - \text{median}(p_0)$. We can also compute this quantization error term for all notes and shift by the median or mean for the melody, or we can use more complicated methods to find the shift which best aligns the sung melody to a grid. We make use of all of these prediction methods in our experiments.

Pitch Clustering Methods

We found that a particularly effective strategy for correcting notes involves shifting the pitch of each note towards the mean pitch of similar notes; we refer to this class of methods as pitch clustering heuristics. These heuristics work well when a particular note is sung more than once in a melody and most of the occurrences of the note are correct. The particular heuristic we use is

$$y_i = \frac{\sum_j I(|\text{median}(p_j) - \text{median}(p_i)| < t)\,\text{median}(p_j)}{\sum_j I(|\text{median}(p_j) - \text{median}(p_i)| < t)}$$

where I is the indicator function and t is a fixed threshold, typically around $.5$. In this equation, the indicator function selects nearby notes, and we average over these notes.

Scale Detection Methods

It is sometimes the case that the intended melody primarily contains notes in a particular musical scale (i.e. C major). If this is the case then it may be possible to detect this scale and predict y_i to be $\text{median}(p_i)$ rounded to the nearest pitch on the scale. We can represent a scale as a root note on the MIDI note number scale and a sequence of offsets specifying pitch values relative to the root note. For example, the C major scale would be represented as 48 (for the root note) and $0, 2, 4, 5, 7, 9, 11$ for the sequence of relative pitch values. In general the root note may not be an integer. A simple approach to scale detection is to choose the scale out of a set of candidate scales which minimizes $\sum_i |\text{median}(p_i) - y_i|^2$ where y_i is $\text{median}(p_i)$ rounded to the nearest pitch on the scale. There are a number of variations including using absolute difference in place of squared difference and adding a term which favors scales with certain note frequency characteristics. We can also vary the set of candidate scales. For example, we can try all possible root notes, only try integer root notes, or only try root notes actually in the sung melody.

We note that the use of scale detection is not always applicable. In some cases the melody may not conform to a particular scale, there may be scale changes in the recording, or—in real-time scenarios—insufficient data may be available for scale detection.

Combining Predictions with Learning

We have described a number of different reasonable methods for predicting the intended note sequence from a transcription of sung pitch. This raises a natural question: which methods should we use? We propose to choose a singer-specific combination of methods using learning. Assume we have a set of methods which predict different values for y_i. With each of these methods, we compute an estimate of the error relative to $\text{median}(p_i)$. We collect all of these pitch error predictions for a particular note into a feature vector x_i. Our approach is to learn a set of singer-specific weights w so that for a note with feature vector x_i our predicted pitch is $\text{round}(w^T x_i + \text{median}(p_i))$. It's helpful to also include a bias feature in x_i which is always set to 1. This lets us learn a singer-specific tuning shift.

The error we ultimately care about is pitch classification error, $I(y_i \neq \text{round}(w^T x_i + \text{median}(p_i)))$. It is hard to minimize this non-convex loss, however, so we instead minimize squared error ignoring the rounding. Given a singer-specific training set consisting of feature vectors x_i and ground truth intended pitch values y_i, we minimize

$$\sum_i (w^T x_i - (y_i - \text{median}(p_i))^2 + \frac{1}{2}\lambda w^T w \qquad (1)$$

where λ is a regularization parameter controlling the norm of the resulting weight vector. This linear least squares objective can be solved very quickly assuming we do not have too

many features. In our experiments, we simply fix our regularization constant $\lambda = 1$. We also experimented with tuning λ on a per-singer basis via cross validation, but we found this did not consistently improve performance. In fact, attempting to tune λ using the relatively small singer-specific training set often hurt performance on the test set.

We have framed the learning problem as simple linear regression, but there are a number of alternative ways of posing it; in experiments not reported here, we tried a number of approaches. These include predicting discrete pitch values using a multiclass classifier, predicting the intended pitch value directly using linear regression, and predicting whether to round the sung pitch up or down. We found that the choice of features is generally more important than the choice of objective and loss function and therefore use a very simple approach which makes it very easy to specify features. We use pitch error as our regression target, as opposed to pitch itself, so the regularization term favors transcriptions close to $\mathrm{median}(p_i)$.

To evaluate our results, we estimate the generalization error of our method by computing leave-one-out cross validation error over melodies. Computing average cross validation error over melodies as opposed to notes ensures that notes in the test data are from different recordings than the training data.

As is often the case for user-specific machine learning, we expect that the total amount of training data collected across all singers will be much greater than the amount of training data collected for any one particular singer. Even if we believe singer-specific training data to be more useful than data from other users, it's still important to also exploit the large amount of extra data from other singers. Methods that use both user-specific and non-user-specific data are common in speech and handwriting recognition where this is called adaptation.

To incorporate this auxiliary data, we use data from other singers to select the set of features for singer-specific weight learning. The objective function we use for evaluating the quality of a set of features is average relative reduction in error. If the leave-one-out cross validation error for a singer is e_1 before singer-specific training and e_2 after singer-specific training, we define relative reduction in error to be $(e_1 - e_2)/e_2$. Average relative reduction in error is this quantity averaged over all users. We use a greedy feature selection strategy to maximize average relative reduction in error: we initially start with an empty set of features. At each iteration we then add the feature which most increases the objective (average relative reduction in error) evaluated using other singers' data. This continues until no new feature increases the objective. The final set of features selected using this method is then used to learn the weight vector w by minimizing Equation 1 on the singer-specific training set.

There are other alternative methods for incorporating data from other singers into singer-specific learning. We also experimented with using data from other singers to learn a weight vector which is used as a prior for singer-specific learning. However, we found the feature-selection-based method to be more robust.

Baseline Transcription Method

Our features use as input a baseline transcription of the sung pitch in the form of a segmented pitch track. Our baseline uses the Praat pitch tracker (Boersma, 2001) with pitch sampled at 100Hz, and our onset detector uses spectral flux and peak-picking heuristics described by Dixon (2006). The onset detector computes spectral flux at 100Hz with a FFT window size of 2048 (46ms at 44kHz).

Each detected onset time is treated as a potential note start time. The corresponding end time is chosen to be the either the first unvoiced frame .1 seconds after the start time or the next detected onset time (whichever comes first). We then throw out all notes that are either shorter than .1 seconds or contain more than 25 percent unvoiced frames (samples that do not have a well-defined fundamental frequency).

In order to establish that our baseline transcription method was comparable to existing approaches, we used Celemony's Melodyne software, a popular commercial system for automatic transcription, to transcribe all of our training data with hand-optimized parameters. We found that our baseline method's output was comparable to Melodyne's output.

To form training and test data for our learning algorithm, we need to match notes in the baseline transcription to notes in the ground truth melodies. Because singers sang along with a drum backing track, the recorded melodies are relatively well-aligned with the ground truth, but there are sometimes extra or missing notes. For each sung note, we find the note in the ground truth melody with the closest start time within a search window. If the ith sung note in a melody has a start time in seconds of s_i, we set the search window for that note to be $[\max(s_i - .4, s_{i-1}), \min(s_i + .4, s_{i+1})]$.

If there are no notes in the ground truth melody within this search window, the sung note is considered unmatched. Unmatched notes are included in the training and test data for the purposes of feature computation, but are not included in error computations or in the data finally used to learn weight vectors. When learning weight vectors we also filter out all notes for which $|y_i - \mathrm{median}(p_i)| > 1$. We assume that these errors are more often than not due to the user forgetting the melody or the key of the melody, and are thus *intention* errors rather than *singing* errors. Finally, as we found that amateur singers are often insensitive to octave and will adjust to the octave most comfortable for their range, we also shift the octave of the ground truth labels for each sequence in order to best match the sung melody.

Experiments

We distributed our data collection application via mailing lists within Microsoft. We received data from 51 participants (17 female). As incentive to participate, users were given a $5 dining coupon upon receipt of their data. The software asked participants to sing 21 examples. Most examples were short recognizable melodies from children's songs (e.g. "Twinkle Twinkle Little Star"), but the set of probes also included some scales and examples with more difficult intervals (e.g. a series of ascending major and minor thirds). With the 21 examples we used, the training procedure took

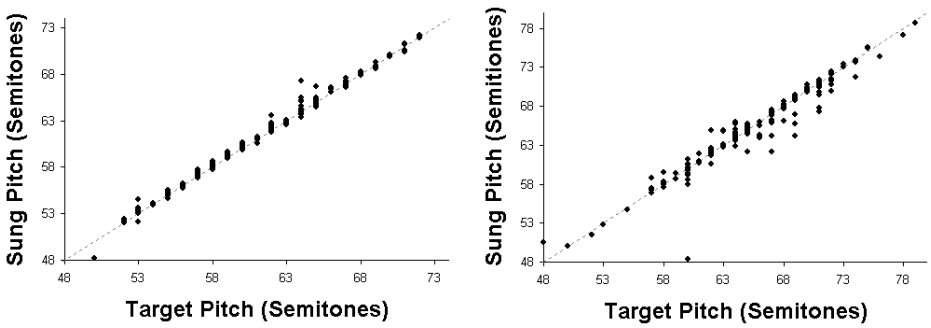

Figure 3: Scatter plot showing target vs sung pitch for notes sung by two different singers. The singer on the left is more accurate, resulting in less spread from the $x = y$ line.

roughly 30 minutes. Audio was recorded uncompressed and normalized and converted to 44kHz, 16-bit mono.

Combining the data from all 51 participants, we had 1071 individual recordings. We used data from 20 randomly selected participants as a development set for designing our features and algorithms and data from the remaining 31 as a test set. We removed data from participants that did not follow instructions or had very poor recording conditions– 5 participants in the development set and 5 in the test set. The results reported here are from this final test set of 26 participants.

We did not specify any skill requirement for participation, and participants varied widely in their singing ability. Some participants reported professional singing experience while others had very little singing experience. Figure 3 shows scatter plots of notes sung by two different singers comparing the sung pitch vs the pitch we asked them to sing. Here we used $\mathrm{median}(p_i)$ as a proxy for the sung pitch of a note. A singer with perfect pitch accuracy would produce a plot with points only within .5 semitones of the $x = y$ line. The first user shown had an average per note accuracy of about 85% while the second user had an accuracy of around 70%.

In our learning experiments we used a set of 38 features consisting of 21 of the "simple" features, 4 pitch clustering features with different threshold values (specifically .25, .5, .75, and 1), 12 different scale-detection-based features, and a bias term. Scale-detection-based features make assumptions about the data and as such are not always applicable, so we are also interested in performance when these features are excluded and report results for this scenario.

Table 1 compares learning results where feature selection and weight learning are performed either with singer-specific data or data taken from other singers. We report average relative reduction in leave-one-out cross validation error using $\mathrm{round}(\mathrm{median}(p_i))$ as a baseline. We define relative reduction in error to be the percent of pitch errors eliminated by training. We used relative as opposed to absolute reduction in error because different singers make very different numbers of mistakes. Both with and without scale detection, the best results came from *singer-specific learning* combined with *non-singer-specific feature selection*. This supports our hypothesis that singer-specific training can give better transcription accuracy.

Figure 4 shows a scatter plot of per user error rates. In this figure it is clear that learning helped and in some cases very significantly reduced error. If we consider only singers with baseline error rates of less than 20%, the average singer had an error rate of 9.45% before training and 5.73% after training. These numbers are for the best method including scale detection features. For many particular singers the effect of learning is even more dramatic. One singer with a 5.60% baseline error rate had an error rate of 0% after training. For a different singer with a 52.99% baseline error rate, learning reduced the number of errors by more than half, to 24.79%.

We also compared our learning method to directly using the non-learning-based pitch prediction methods. With scale detection methods, the best of these methods gave a 24.46% average relative reduction in error on the test set. Excluding scale detection, the best method gave a 13.45% average relative reduction in error. In both cases, our learning method outperformed the non-learning-based method despite selecting the non-learning method with best relative error reduction on the entire data set (i.e. fit to the test data).

Finally, we also tried some other variations of non-singer-specific learning to see if they could beat singer-specific learning. We found by tuning λ to minimize leave-one-out cross validation error on the training set, we could slightly improve the non-singer-specific learning results to 22.29% (from 22.08%) in the scale detection case and 17.16% (from 16.82%) in the w/o scale detection case. However, we could not find a method that that outperformed singer-specific training.

Real-Time Transcription

In the experiments discussed so far, we assumed when computing features and predictions for a particular sung note that we had access to the entire recording, including notes sung after the note in question. In real-time performance applications, however, this is not the case. To simulate learning in these scenarios, we also tried computing features for each note using a truncated version of the baseline transcription with only limited look-ahead. Figure 5 shows these results for different look-ahead window sizes. In this figure, the top line shows singer-specific weight learning with non-singer-specific feature selection, and the bottom line shows non-singer-specific weight learning. Scale detection fea-

Average Relative Error Reduction With Scale Detection		
	Singer-Specific Weights	Non-Singer-Specific Weights
All Features	24.71	22.08
Singer-Specific Features	26.23	20.23
Non-Singer-Specific Features	**27.22**	21.55
Average Relative Error Reduction Without Scale Detection		
	Singer-Specific Weights	Non-Singer-Specific Weights
All Features	13.96	15.62
Singer-Specific Features	15.84	13.73
Non-Singer-Specific Features	**19.83**	16.82

Table 1: Learning results with and without scale detection features. Higher values correspond to more accurate transcription.

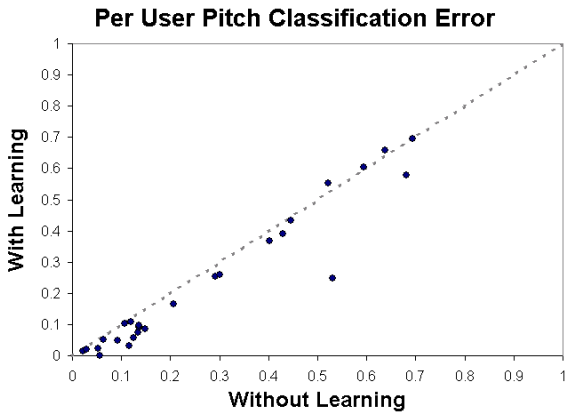

Figure 4: Scatter plot showing error with learning vs error without learning. Points below the $x = y$ line represent an improvement through learning. This figure demonstrates that pitch classification is improved through learning for most singers in our study (22 out of 26). These results use non-singer-specific feature selection combined with singer-specific weight learning. Here we include scale detection features; without scale detection, errors are slightly increased but the benefit of learning remains.

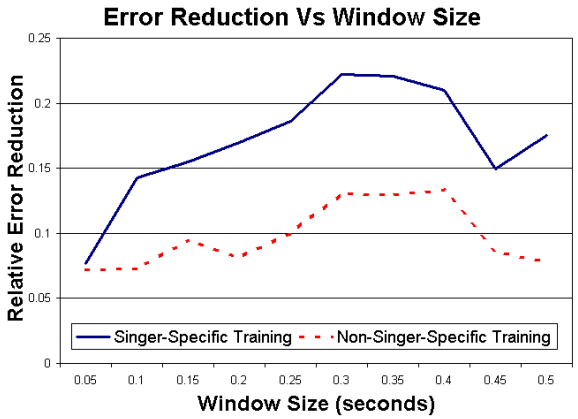

Figure 5: Error reduction as a function of the look-ahead data window available to the classifier. Results show that training can improve transcription even for very short window sizes, with singer-specific training reducing error more.

tures were not used in either case, as this is likely to be impractical for real-time scenarios. We also tried non-singer-specific weight learning with feature selection, but this performed worse. As seen in Figure 5, singer-specific learning again outperformed non-singer-specific learning and gave a significant reduction in error over the baseline method. We note these experiments are only meant to roughly approximate the difficulties in real-time performance applications since in fact our pitch tracking, onset detection, and note segmentation algorithms still take as input the entire recording. We leave a full study of real-time transcription as future work.

Related Work

Our work is distinct from previous automatic transcription work in that we focus on singer-specific training for transcribing *intended pitch* as opposed to *sung pitch*. Ryynänen (2006) present a good general overview of the vocal melody transcription problem, and Clarisse et al. (2002) compare a number of systems for transcribing sung as opposed to intended pitch. We know of only one previous paper that has considered singer-specific training for transcription (Weihs and Ligges, 2005). In that work, the authors show that by tuning the parameters of their transcription system on a singer-specific basis they can achieve better performance on a small data set of 7 singers. These parameters control the details of the fundamental frequency estimation method and a pitch smoothing heuristic to account for vibrato. In our work, we propose a method for automatically selecting and combining multiple different pitch prediction methods, and we evaluate singer-specific training on a data set with many more singers.

Little, Raffensperger, and Pardo (2008) use singer-specific training for a query-by-humming task. In this application, the goal is to match the sung melody to a database of melodies. In contrast, in our application the goal is to transcribe the intended melody. We do not assume the availability of a database of melodies, and in fact we assume that the intended melody may be completely novel. The training method used by Little, Raffensperger, and Pardo (2008) tunes the parameters of their note segmentation algorithm and also learns a similarity measure for musical intervals which is used to align pairs of melodies. Meek and Birm-

ingham (2004) also consider a trainable model of singing errors for query-by-humming but do not specifically consider singer-specific training.

Several authors have used non-singer-specific learning to improve transcription systems. Ryynänen (2004) learn a model of the pitch contour of a note to improve vocal melody transcription transcription and also use scale detection with probabilistic models of note transition probabilities. Ellis and Poliner (2006) use a classifier to extract the predominant melody in a polyphonic audio recording. The classifier is trained on low-level audio features and the challenge is separating the melody from other instruments.

Several authors have also proposed different methods for accounting for constant and drifting tuning errors made by singers (Haus and Pollastri, 2001; Wang, Lyu, and Chiang, 2003; McNab, Smith, and Witten, 1996; Ryynänen, 2004). These methods are in some cases similar to the features we use and could potentially be incorporated in our method. Finally, we note there are many commercially available systems for automatic transcription from audio. We found Celemony's Melodyne software to be, subjectively, the most reliable system of those we evaluated. We in fact found that many available programs do not give reliable transcriptions on real-world voice recordings.

Discussion

We feel that some of techniques we developed for singer-specific pitch tracking will be applicable to other domains as well. Many situations requiring learning and tight coupling between the user and machine would be amenable to our procedure, particularly when a user is controlling a computer via a noisy input, e.g. controlling a video game character through electrophysiological sensors. We expect that any such scenario will require the careful design of an interactive data collection procedure in order to capture the true variations in human performance, as well as a flexible learning mechanism to integrate existing heuristics and features along with newly proposed features. Finally, given the small amount of data that will likely be available for each individual, it will be important to consider the best ways in which to integrate information from the individual and the ensemble when training the learner. We hope that the solution we have developed and presented here will help in such related scenarios.

Supplementary Material

A portion of our data set, including all of our ground truth recordings and a more complete description of our experimental procedure, is available at `http://research.microsoft.com/cue/pitchtracking`.

References

Boersma, P. 2001. Praat, a system for doing phonetics by computer. *Glot International* 5(9).

Clarisse, L. P.; Martens, J. P.; Lesaffre, M.; Baets, B. D.; Meyer, H. D.; Demeyer, H.; and Leman, M. 2002. An auditory model based transcriber of singing sequences. In *ISMIR-02*.

Dixon, S. 2006. Onset detection revisited. In *DAFx-06*.

Ellis, D. P., and Poliner, G. E. 2006. Classification-based melody transcription. *Machine Learning* 65(2-3):439–456.

Haus, G., and Pollastri, E. 2001. An audio front end for query-by-humming systems. In *ISMIR-01*.

Little, D.; Raffensperger, D.; and Pardo, B. 2008. User Specific Training of a Music Search Engine. In *Machine Learning for Multimodal Interaction*.

McNab, R. J.; Smith, L. A.; and Witten, I. H. 1996. Signal processing for melody transcription. In *19th Australasian Computer Science Conference*.

Meek, C., and Birmingham, W. 2004. A comprehensive trainable error model for sung music queries. *Journal of Artificial Intelligence Research* 22(1):57–91.

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

Ryynänen, M. 2006. Singing transcription. In Klapuri, A., and Davy, M., eds., *Signal Processing Methods for Music Transcription*. Springer Science. 361–390.

Wang, C.-K.; Lyu, R.-Y.; and Chiang, Y.-C. 2003. A robust singing melody tracker using adaptive round semitones (ars). In *ISPA-03*.

Weihs, C., and Ligges, U. 2005. Parameter optimization in automatic transcription of music. In *Conference of the Gesellschaft für Klassifikation*.