Generating Emotionally Relevant Musical Scores for Audio Stories

Steve Rubin
University of California, Berkeley
srubin@cs.berkeley.edu

Maneesh Agrawala
University of California, Berkeley
maneesh@cs.berkeley.edu

ABSTRACT

Highly-produced audio stories often include musical scores that reflect the emotions of the speech. Yet, creating effective musical scores requires deep expertise in sound production and is time-consuming even for experts. We present a system and algorithm for re-sequencing music tracks to generate emotionally relevant music scores for audio stories. The user provides a speech track and music tracks and our system gathers emotion labels on the speech through hand-labeling, crowdsourcing, and automatic methods. We develop a constraint-based dynamic programming algorithm that uses these emotion labels to generate emotionally relevant musical scores. We demonstrate the effectiveness of our algorithm by generating 20 musical scores for audio stories and showing that crowd workers rank their overall quality significantly higher than stories without music.

Author Keywords

Audio stories, storytelling, musical scores, music retargeting

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

The tradition of oral storytelling is thousands of years old and still endures today through digital recordings like audiobooks and podcasts. At audiobook sites like Audible [2] and LibriVox [13], these recordings number in the hundreds of thousands. In 2013, users downloaded over 500 million hours of audiobooks from Audible alone [2], nearly 15% of Americans listened to an audiobook [37], and 14% of Americans listened to podcasts every month [32].

Highly-produced audiobooks, podcasts, and children's stories combine a speech recording, which vocally conveys the story, with a musical score that serves to emphasize and add nuance to the emotions of the speech. Yet, crafting such a musical score involves smoothly re-sequencing, looping and timing the music to match the emotions in the story as they change over the course of the narrative. This process requires significant audio production expertise and is challenging even for expert producers. As a result, most of the recorded stories available today focus on providing the speech track and do not contain a musical score.

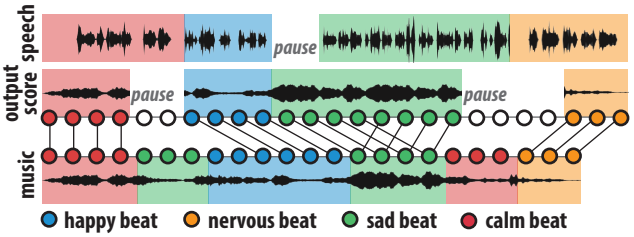


Figure 1. Our algorithm re-sequences the beats (circles) of the input music (bottom row) to match the emotions of the speech (top row). Our algorithm inserts pauses in speech and music, and makes musical transitions that were not in the original music in order to meet these constraints.

In this paper we present a system that allows users with no audio editing expertise to generate high-quality, emotionally relevant music scores for audio stories. To generate a score, our system requires a speech track with associated emotion labels, and one or more music tracks with associated emotion labels. Our system uses a dynamic programming optimization approach to re-sequence the input music tracks so that the emotions of the output score match those of the speech (Figure 1). We studied how expert producers use music in audio stories and incorporate additional constraints into the optimization that correspond to structural and stylistic design decisions in their scores.

We investigate three techniques for labeling the speech and music. We provide a manual interface for hand-labeling speech emotion by annotating a text transcript, and for labeling music emotion by listening to and annotating segments of the music. We also provide a crowdsourcing method for generating the labels, where crowd workers use the labeling interface and our system combines their responses. Finally, our system includes a fully automatic method for computing the emotion labels for the speech and music using sentiment analysis techniques. These three approaches offer different trade-offs in time and label quality that affect the overall quality of the music scores that our system generates.

We demonstrate the effectiveness of our score generation tools by generating a total of 20 musical scores including examples using each of our three labeling methods. We ask listeners to rank the overall quality of these results. Surprisingly, listeners rank the results generated with our crowd-

UIST 2014, October 5-8, 2014, Honolulu, HI, USA.
ACM 978-1-4503-3069-5/14/10.
http://dx.doi.org/10.1145/2642918.2647406

labeling technique as high or higher than the hand-labeled results, which are in turn better than results using our automatic approach and no music. Our system generates scores that are preferable to no musical score for all of the speech tracks. We also investigate the inter-labeler reliability of agreement and find that human-produced labelings tend to agree more with each other than with our automatic labelings.

RELATED WORK

Automatically scoring audio and video stories is a longstanding problem in multimedia research. Foote et al. [9] automatically create music videos by editing video clips to match a piece of music. Their method finds suitable locations to cut video and then splices video to match points in music where audio features change. Our work instead edits the music to create a soundtrack tailored to the speech by matching emotions rather than low-level audio/video features. Monteith et al. [18, 17] generate new music MIDIs to match automatically-labeled emotions in stories. They learn generative models of different music emotions and then sample new music from these models. However, because their algorithm only considers local matching of speech and music, the soundtracks they generate may not contain story-level structure or cohesion. Our algorithm uses the emotion labels across an entire story to generate a globally coherent musical score. We generate emotionally relevant musical scores by re-sequencing existing, high quality music tracks instead of MIDIs, and we provide more options for emotion labeling.

Rubin et al. [22] focus on the process of editing speech and adding music to audio stories. Their system offers a high-level editing workflow and requires a user-in-the-loop to refine speech and music edits. Our system asks for even higher-level input and does not require a user to specify or refine edits directly. Earlier work from Rubin et al. [21] allows producers to generate a musical score for a short section of speech, while our system creates an optimal musical score for an entire story. Neither of these systems deal directly with matching the emotions of speech and music. Our system allows novices to create emotionally relevant musical scores for audio stories.

Dynamic programming is a common strategy for concatenative music synthesis, where short music clips are re-sequenced from a database to create new music that mimics existing music, or to create music that follows certain note patterns [28, 27] Our work follows this general approach, but we offer constraints that generate emotionally relevant music scores rather than music that imitates other music. Wenner et al. [34] and Rubin et al. [22] present methods for music retargeting to find optimal paths through music that match constraints of videos or audio stories. Our work builds on these algorithms and describes new constraints to match emotions of the speech and music, add pauses with flexible length, bound music segment lengths, and create scores from multiple music tracks. Other work applies belief propagation [29] and genetic algorithms [33] to music re-sequencing, but we use dynamic programming instead to efficiently find optimal solutions. Algorithms for video synthesis often apply dynamic programming techniques as well, as in Schödl et al.'s [26] video textures, and Arikan et al.'s [1] annotation-based character motion synthesis.

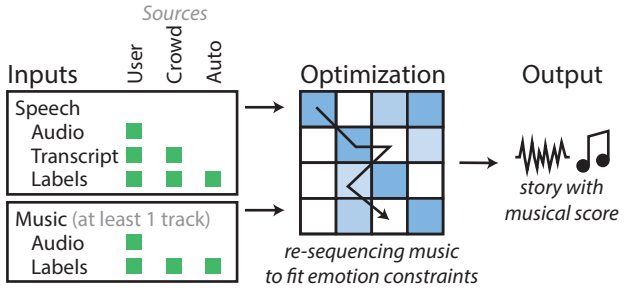


Figure 2. Our automated musical score generation system requires a speech track and music tracks, and emotion labels on those tracks. Our algorithm re-sequences the music tracks to create an output musical score that matches the emotions of the speech. The green boxes denote the sources—user, crowd, and fully automatic—that our system provides for obtaining each input. Our system also requires a speech transcript so users and crowd workers can more quickly label the emotions by reading the text instead of listening to the speech.

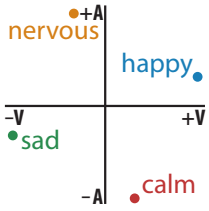
Affect prediction is a well-studied problem for speech [6], text [4], and music [11]. We draw on these techniques to provide automatic emotion labeling tools in our system. Our work concentrates on the application of affect labeling and understanding rather than on its prediction.

LABELING EMOTIONS

Figure 2 shows our pipeline for generating emotionally relevant musical scores. A user begins by selecting a speech track such as an audiobook, and one or more instrumental music tracks (i.e., music without lyrics). Our system then gathers emotion labels for each of the speech and music tracks. After collecting these labels, our system constructs a musical score by re-sequencing the input music tracks so the emotion labels of the output music match the emotion labels of the speech.

Our system offers three methods for obtaining the required emotion labels: hand-labeling, crowd-labeling, and automatic labeling. These three methods offer trade-offs in time, effort, and personalization. Hand-labeling produces emotions that reflect the user's emotions and may result in the most personal musical score, but it takes the most time for the user. The crowd-labeling method requires no extra work on the user's part and incorporates human ratings. However, this method takes more time than hand-labeling to acquire emotion labels. Finally, the fully automatic method produces labels immediately, but they may not accurately reflect the personal emotions that the user—or any human—feels about the story and the music. As we will show, all of these methods require a speech transcript, which may cost money to obtain. The crowd-labeling method has the added cost of paying workers.

Affect researchers have found Russell's circumplex model [23] of quantifying emotion with 'valence' and 'arousal' to be highly consistent with behavioral and cognitive neuroscience study results [20]. The valence dimension indicates whether an emotion is positive or negative, whereas arousal indicates the intensity of the emotion. A user does



not need to learn the circumplex model. Instead, we focus on four emotions, *happy, nervous, sad,* and *calm*, because they almost evenly span the circumplex [23]. As shown in the inset, we set their coordinates to $(.95, .31)$, $(-.31, .95)$, $(-.95, -.31)$, and $(-.95, .31)$, respectively, an equal distribution near their original locations [20].

Speech Emotion Labels

Our goal is to break the speech into segments of similar emotions. Segmenting a speech track into emotionally similar segments based solely on the raw waveform requires careful listening and can be difficult. In text, however, paragraphs usually represent topically-coherent segments that convey one predominant emotion. Therefore, we obtain a text transcript of the speech and use its paragraph boundaries to segment the speech.

Audiobooks and even some podcasts often have readily available transcripts, but if not, the user can obtain one a from crowdsourced transcription site like CastingWords [5] for $1.00 per minute of audio. We time-align the text transcript and the speech using a variant of the Penn Phonetics Lab Forced Aligner [36] from Rubin et al [22]. The time-alignment allows our system to find the segment of speech that corresponds to each paragraph of the text.

The user chooses one of the three labeling methods to get emotion labels for each speech segment.

Hand-labeling the speech. If users wish to personalize the emotion labels of the speech, they can label the emotion of each paragraph of the text transcript by hand (Figure 3).

Crowd-labeling the speech. To crowd-label the emotions of the speech, our system posts tasks to Amazon's Mechanical Turk. These tasks are identical to the hand-labeling interface (Figure 3).

Crowd workers often try to complete tasks as quickly as possible so they can maximize their hourly rate. Such workers may not fully engage with the speech or its emotions and produce inaccurate emotion labels. Yet, emotions labels can be subjective, and there is no way to test the accuracy of a worker's labeling. Our approach is to obtain emotion labels from multiple workers and then select the labeling of a single worker that best represents all of the workers.

We interpret each worker's complete speech labeling as a sample from the distribution of possible speech labelings, first assigning each paragraph i a distribution p_i where $p_i(l)$ is the probability that a worker labeled paragraph i with emotion l in all of the labelings we have collected. For example, if we have ten workers total and 7 label paragraph 1 as 'sad' and 3 label it as 'calm'—then $p_1(sad) = 7/10$, $p_1(calm) = 3/10$, and $p_1(happy)$ and $p_1(nervous)$ are both zero. We then find the worker that gave the most probable labeling according to this distribution. Suppose $\ell = \{l_1, \ldots, l_k\}$ is a worker's labeling of the sequence of paragraphs in the speech. We compute the probability of this labeling as

$$P(\ell) = \prod_{i=1}^{k} p_i(l_i).$$

Our system computes this probability for each worker's labeling and assigns the labeling with the highest probability to the speech. An alternative to our *most probable worker labeling* approach is to choose the most frequently labeled emotion for each paragraph independently. However, such a voting scheme can yield emotion transitions between paragraphs that did not appear in any of the individual worker's labelings. In contrast, our approach yields a labeling that is guaranteed to be consistent with at least one worker's labeling. This strategy of picking one representative rather than averaging or aggregating worker results has been used in recent work to ensure self-consistent results [12, 35].

Automatically labeling the speech. To automatically label the speech, our system estimates the emotion of each paragraph based on the emotion conveyed by each word. Warriner et al. [31] have collected a corpus of crowdsourced valence and arousal ratings for nearly 14,000 English words. We normalize these scores by the global valence/arousal mean and standard deviation. We then compute the average of the normalized valence/arousal scores of all words in a paragraph. Our system projects those averages to the nearest of our four labels to obtain an emotion label for each paragraph.

Music Emotion Labels

Our system provides techniques for music emotion labeling similar to our speech labeling methods. As in speech labeling, we first break the music into segments. Structural segments of music often contain one predominant emotion because the features that differentiate structure—timbre, pitch, volume, and self-similarity—are also indicative of music emotion. Following McFee and Ellis [15], our system segments music by computing a hierarchical clustering of self-similarities in a track and finding an optimal pruning of the cluster tree.

The user selects one of the three labeling methods to get emotion labels for each music segment.

Hand-labeling the music. If users wish to personalize the emotion labels of the speech, they can listen to and assign labels for each music segment (Figure 3) As in hand-labeling the speech, this method is preferable for users that have time and desire a personalized musical score.

Crowd-labeling the music. To crowd-label the emotions of the music, our system asks workers on Mechanical Turk to listen to and label the emotions of the music segments for an entire track (Figure 3). Our system then selects a final emotion labeling by finding the worker's labeling that best represents all of the worker labelings (see earlier section on *Crowd-labeling the speech*).

Automatically labeling the music. Schmidt et al. [24, 25] have developed methods for automatically predicting the valence and arousal of music. Their MoodSwings Turk dataset consists of a large set of crowd-generated, per-second valence/arousal labels and accompanying audio signal processing features (MFCCs [14], spectral contrast [10], and chroma [7]). Our goal is to automatically predict the emotion of each music segment, but the dataset contains per-second labels and no notion of segments. We follow Schmidt et

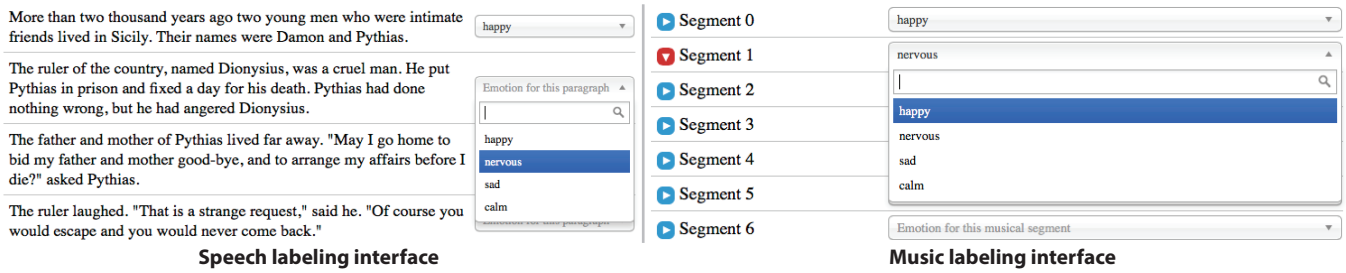


Figure 3. Our interface for labeling speech emotion (left) asks the labeler to annotate each paragraph with an emotion. Our interface for labeling music emotion (right) asks the labeler to label each music segment with an emotion.

al. [24] and train a multiple linear regression model on the MoodSwings Turk dataset to predict per-second emotions.

Emotion in music has a time dependency. That is, emotions at times before time t influence the emotion at time t. To account for this dependency, our model uses ten seconds (times $t-9,\ldots,t$) of per-second MFCC features to predict the valence and arousal at time t [24]. Our system predicts the valence/arousal at each second of each music segment and then finds the average of the predictions. Because of a limited corpus of music in the training set, this model often reports that all segments of a track have the same emotion whereas human labelers give more varied labelings. We subtract the overall segment average from each music segment's average to get a new, more varied prediction. Then, we project each prediction to the nearest of our four emotion labels to get a label for each segment.

SCORE GENERATION ALGORITHM

We designed our score generation algorithm to generate musical scores that match the emotion of the speech while also following other characteristics of high quality musical scores. We identified these characteristics by listening to hours of expertly-produced audio stories, noting structural and stylistic patterns in their musical scores. We further studied books and online sources on audio story production to refine these characteristics. Figure 1 shows an example of how our algorithm re-sequences input music to match the emotions of the speech. To generate an emotionally relevant score, our algorithm translates emotion matching and our other constraints into numeric costs. The two primary costs are *matching costs*—costs of how well the music emotions match the speech emotions—and *transition costs*—costs for beat-to-beat transitions in the music. Once our algorithm computes these costs, it searches for the lowest cost musical score using dynamic programming.

We specify our optimization and constraints on beats, which are short, structural units of time in music. Formally, we consider a piece of music as a set of n beats $\mathbb{B}=\{b_1,b_2,\ldots,b_n\}$ where $b_1,b_2,\ldots,b_n$ is the natural order of beats in the music. To detect the beats in a piece of music we apply Ellis' beat tracking algorithm [7]. Beats lengths within a song vary because tempo often shifts within a music track, so we use the average beat length as our unit of time throughout the algorithm. Next we compute m, the number of music beats required to score the speech. We set m to the duration of the speech track divided by the average beat length of the music. Our algorithm searches for a sequence $\boldsymbol{s}$ of m beats that minimizes the matching and transition costs; it then generates the output musical score by re-sequencing the original beats according to $\boldsymbol{s}$. Figure 1 shows an example of re-sequencing music beats (bottom row of beats) to match story emotions and duration (top row of beats).

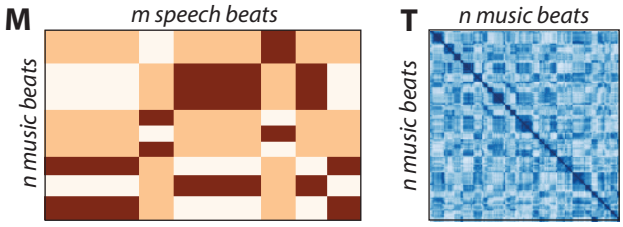


Figure 4. The matching cost table M (left) has lower cost when music beat b_i and speech beat k have the same emotion. The transition cost table T (right) gives the cost of moving between beat b_i and beat b_j. These examples show transition and matching cost tables with only the transition and matching constraints. Darker colors imply cheaper costs.

Tables M with size $n\times m$ and T with size $n\times n$ store the transition and matching costs, respectively (Figure 4). Entry $M_{i,k}$ is the cost for playing beat b_i at speech beat k in the output score. The matching cost encodes how well the music emotion fits the speech emotion. Entry $T_{i,j}$ is the cost for moving from beat b_i to beat b_j and is independent of the location in the output score. The transition cost encodes whether the music could make a seamless-sounding jump from b_i to b_j. Our algorithm searches for the sequence of beat indices $\boldsymbol{s}=[s_1,\ldots,s_m \,|\, s_i\in\{1,\ldots,n\}]$ of m beats that minimizes

$$cost(\boldsymbol{s})=\sum_{k=1}^{m} M_{s_k,k}+\sum_{k=1}^{m-1} T_{s_k,s_{k+1}} \qquad (1)$$

where k is the speech beat index in the output score. A recursive form of this equation enables us to apply dynamic programming to find the optimal beat sequence. So, the recursive minimum cost of a length-m sequence $\boldsymbol{s}$ ending with beat b_j is

$$c(j,m)=\min_{i\in\{1,\ldots,n\}}\left(c(i,m-1)+M_{j,m}+T_{i,j}\right).$$

The right-hand side is a minimum over three terms. The first term, $c(i,m-1)$ is the minimum cost $(m-1)$-length sequence that ends in beat b_i. The second and third terms, $M_{j,m}+T_{i,j}$ are the matching and transition costs for the last

beat b_j. The optimal sequence $\boldsymbol{s}$ with any ending beat then has cost

$$\min_{j \in \{1,\ldots,n\}} c(j, m).$$

Because we have expressed the minimum cost sequence recursively, in terms of its subproblems, we can apply dynamic programming to find the optimal beat sequence.

Matching costs

The main purpose of our system is to create music that matches the emotions of the accompanying speech. The speech and music contain emotion tags (happy, nervous, sad, and calm) and their respective numerical values in valence/arousal space. We set matching costs in table M so the valence and arousal of the optimal output score matches the valence and arousal of the speech as closely as possible.

To encode this constraint in our optimization, we compute the ℓ_2 distance $D_{va}(i, k)$ between speech valence/arousal and music valence/arousal at each music beat b_i and speech beat k. We set matching cost $M_{i,k} = w_{va} D_{va}(i, k)$, where w_{va} is a constant that controls the importance of the matching cost in our optimization.

Transition costs

A transition from beat b_i to another beat b_j sounds natural when the timbre, pitch, and volume of b_j are similar to the timbre, pitch, and volume of b_{i+1}, the next beat in the original beat sequence. We compute the transition cost table T by combining timbre, pitch, and volume distances (D_t, D_p, and D_v) for each pair of beats:

$$T_{i,j} = w_t D_t(i+1, j) + w_p D_p(i+1, j) + w_v D_v(i+1, j)$$

Our algorithm estimates timbre differences by computing MFCCs [14] for each beat, and then computing the cosine distance $D_t(i, j)$ between MFCCs for pairs of beats b_i and b_j. We similarly estimate pitch differences by computing chroma features [7] for each beat, and then computing the cosine distances $D_p(i, j)$ between beat pairs.

Two beats that are similar in pitch and timbre may not be similar in volume. A large volume difference between beats can create a jarring listening experience. We compute the RMS energy, a measure of volume, of each beat and then take the absolute energy difference between all beat pairs (b_i, b_j) to get $D_v(i, j)$.

The weights w_t, w_p, and w_v allows us to control the acceptable range of timbre, pitch, and volume differences in beat transitions. These transition cost constraints ensure that the original beat progression—a natural sounding progression—has zero transition cost (i.e., $T_{i,i+1} = 0$).

Structural constraints

Our algorithm can generate musical scores that match the emotions of speech using only the basic matching cost and transition cost constraints. However, high-quality scores contain additional structure by using pauses and limiting music segment lengths.

Pauses. In most high-quality audio stories, the music is not constantly playing. Instead, music occasionally fades in and out to create sections of the story without music. These pauses call attention to the fact that the musical score exists, forcing listeners to think about the music and its emotions.

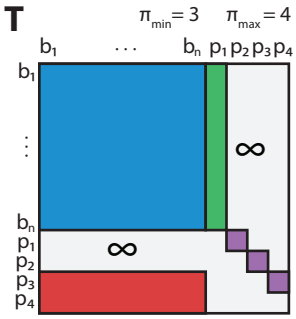


Figure 5. Our algorithm enables pauses in the musical score by extending the transition cost table T with pause beats.

Figure 5 shows how we adjust our transition cost table T to accomodate pauses in a musical score. We first define π_{min} and π_{max}, the minimum and maximum beat lengths of pauses, respectively. We concatenate π_{max} rows and columns to the original T (the blue block in Figure 5 is the original T). The π_{max} new beats are pause beats. Each of these added beats p_i represents the ith beat of a pause.

Every pause in music starts at the first pause beat; any beat in music can transition to beat p_1 but no other pause beats (green rectangle in Figure 5). To maintain the invariant that p_i is the ith beat of a pause, we only allow p_i to transition to p_{i+1} (purple squares). Any pause beat at or above the minimum pause length π_{min} has a free cost transition back to any music beat (red rectangle). In practice, we assign a cost κ_{start} to entering a pause (green rectangle) that is greater than the cost of a natural-sounding music transition (see *Setting Parameters*, below). This cost ensures that our score does not add pauses when it can play good-sounding music. We also add a small cost of κ_{intra} to transitions between p_i and p_{i+1} if $i > \pi_{min}$. Without this cost, most pauses would have length π_{max} because the intra-pause transitions would be free while optimal sequences in music often contain non-free transitions.

Music segments. High-quality audio stories tend not to play overly short or long segments of music. Short segments do not give the music time to integrate with the speech and its emotions, while overly long segments can cause the listener to "tune out," or ignore the music. Our algorithm constrains the length of music segments in the output score.

Figure 6 shows how we constrain the length of a music segment to at least δ_{min} beats and at most δ_{max} beats. In order to keep track of the lengths of music segments, we create a new transition cost table T'. Each index in the table represents a (beat, segment-length) pair (b_i, d_k) where segment length d_k represents the kth beat in the current music segment. We construct this new table by copying blocks from the existing transition cost table (the colored blocks from Figure 5 correspond to the colored blocks in T' in Figure 6).

To maintain the invariant that a beat a segment length d_k is the kth beat of the music segment, we only allow (b_i, d_k) to transition to (b_j, d_{k+1}) for any beats b_i and b_j (blue squares in Figure 6). Once the music has been playing for δ_{min} beats, a pause can begin. Any (beat, segment-length) pair that is at or above the minimum segment length δ_{min} can transition to

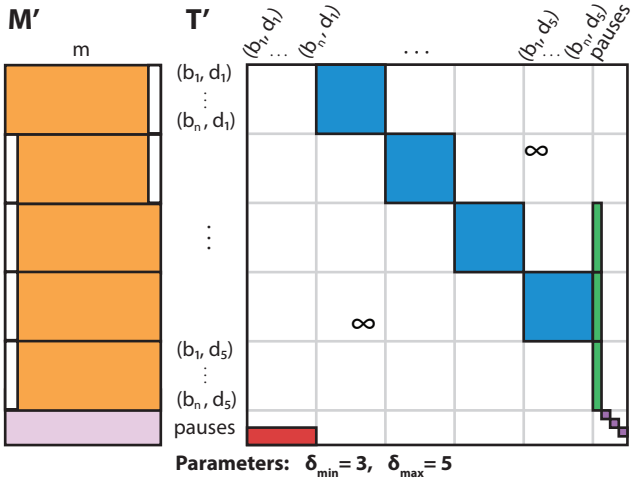


Figure 6. We provide constraints on the length of music segments by creating a new matching cost table M' and transition cost table T'. The indices of the new tables are (beat, segment-length) pairs rather than just beats. We build the M' table from δ_{max} copies of M, except that the music must start at beat-length d_1 and end at beat-length greater than δ_{min}. We copy blocks of table T in Figure 5, indicated by colors, to create table T'.

the first pause beat p_1 (green rectangles). We copy the intra-pause transitions directly from T (purple squares). Finally, a pause must transition back to a (beat, segment-length) pair at segment length d_1, because all music segments start with length 1 (red rectangle).

We also create a new M' with constraints for music segment lengths. The first (beat, segment-length) pair of the output score must be at music segment length 1. Otherwise, we could not enforce the minimum length on the first music segment in the output score. The last (beat, segment-length) pair of the output score must be at music segment length at least δ_{min}, so the last segment of the score is longer than the minimum segment length. We set the matching costs to infinity where these conditions do not hold (see M' in Figure 6).

Stylistic constraints

Expertly created scores employ specific stylistic techniques in order to improve the overall score quality. We have designed constraints to imitate these techniques.

Minimum loop constraint. High-quality musical scores often contain loops to allow a short section of music to score a longer section of speech, but if these loops are too short, the music can sound unnaturally repetitive. We introduce a minimum loop constraint to prevent the score from looping sections of music that are less than eight beats long by setting $T_{i,j} = \infty$ if $i - 8 < j \leq i$.

Musical underlays. Expert producers often add *musical underlays* to audio stories, aligning changes in structure, timbre, and volume of music with important points of emphasis in speech. After the music change, the speech pauses for a short period of time and the music plays solo. Rubin et al. [21] describe the benefits of adding musical underlays to audio stories.

To generate musical underlays in our optimization, we first preprocess the input speech, adding 6 second pauses at all

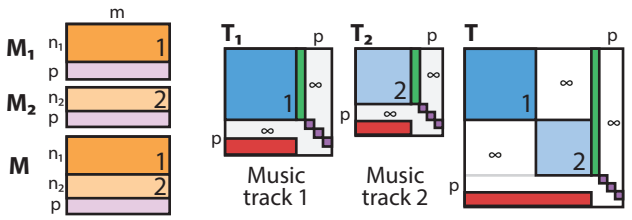


Figure 7. Our algorithm generates scores composed of multiple music tracks. It first computes tables M (left) and T (right) for each music track using a consistent speech beat unit, and then combines the tables to form the final M and T. This construction of T does not allow inter-song transitions, but does allow for efficient optimization.

emotion changes—emphasis points often occur at emotional changes in the speech, so these are the candidate locations for musical underlays.

We then modify the matching table M to align music change points with the speech emphasis points we added in the pre-processing step. To do this, our algorithm first finds the set of change point beats in the music using a novelty-based change point detection algorithm [8, 22]. This algorithm computes a similarity matrix of per-beat music features, and then identifies large changes on the diagonal, which indicate significant differences in the music features. We examine the music segment on either side of each change point to identify change points that correspond with transitions in emotion labels. Our transition cost table gives a large preference to playing music change points in the output score when emotion changes in music match emotion changes at speech emphasis points. If b_i is a change point whose emotion change matches the emotion change at speech beat k, we increase $M_{j,k}$ by constant $\kappa_{changepoint}$ for all $j \neq i$. This effectively penalizes all other music beats when a change point music beat matches the speech emotion change.

Another strategy expert producers use to emphasize points in the speech is to start or stop the music at those points. To apply this strategy, we penalize music stopping and starting at points other than emotion change boundaries. If speech beat k is not an emotion change boundary, we add cost κ_{exit} to $M_{p_1,k}$ and κ_{enter} to $M_{p_i,k}$ for $i \geq \pi_{min}$. This approach lowers the relative cost for starting and stopping music at emotion changes in the story.

After the algorithm has generated the optimal score, some of the 6 second speech pauses we added in preprocessing may align with pauses in the musical score. Rather than playing six seconds of silence, our algorithm contracts the final audio, deleting regions where a pause in speech aligns with a pause in the music.

Multiple music tracks. Musical scores often contain segments from multiple music tracks, especially for longer stories. Moreover, most music tracks do not contain every emotion, so using multiple tracks increases the likelihood of matching the output score to the speech emotions.

Figure 7 shows how our algorithm combines transition and matching cost tables from two songs. We insert the music beat transition costs from each music track's T along the diagonal (blue squares in Figure 7. Music beats can only tran-

Speech tracks					Generated musical scores		
Speech name	Author	Narrator	Dur.	Source	Hand labeled	Crowd Labeled	Auto Labeled
Damon and Pythias	Ella Lyman Cabot	Ginger Cuculo	1:38	LibriVox	**alice**	**alice**, 2815ad, highschool, tub, learn	**alice**
Not That I Care	Molly Reid	James Wood	2:38	NPR	**2815ad**	**2815ad**, alice+timemachine	**2815ad**
Roosts	Zach Brockhouse	Michael Cunningham	3:12	NPR	**tub+highschool**	**tub+highschool**, intriguing+alice	**tub+highschool**
Goldilocks		LearnEnglish Kids	2:10	YouTube	**learn**	**learn**, timemachine	**learn**
The Story of an Hour	Kate Chopin	Matt Bohnhoff	7:20	LibriVox		intriguing+learn	

Table 1. We generated 20 scores spanning five different audio stories. The short names on the right correspond to music in Table 2. Crowd listeners evaluated the musical scores in bold. A '+' represents a musical score generated with two tracks using our multiple tracks constraint.

Short name	Music track	Artist
alice	Alice Returns	Danny Elfman
2815ad	2815 A.D.	Thomas Newman
highschool	Highschool Lover	Air
learn	You Learn	Jon Brion
tub	The Bathtub	Dan Romer
timemachine	Time Machine	Ryan Miller
intriguing	Intriguing Possibilities	Trent Reznor & Atticus Ross

Table 2. We used seven different instrumental music tracks in the musical scores we generated.

sition to other beats within the same music track or to pause beats.

After a pause ends, either music track can play (red rectangle, Figure 7). We stack the two matching cost tables to create a new matching cost table that covers all possible beats in the musical score (orange rectangles, Figure 7). The running time of our algorithm increases linearly in the duration of music added, instead of quadratically, because it does not need to consider inter-music-track transitions.

Setting parameters

We set the main structural parameters based on listening to audio stories: π_{min} of 20 seconds, π_{max} of 35 seconds, δ_{min} of 20 seconds, and δ_{max} of 90 seconds.

Next, we set the distance weights in our optimization empirically, based on the importance of the constraints. We set the acoustic distance parameters $w_t = 1.5, w_p = 1.5$, and $w_v = 5$—volume differences are small in magnitude relative to timbre and chroma cosine distance. We set the emotion matching distance parameter $w_{va} = 1$.

Finally, we set the cost parameters: the cost for entering a pause $\kappa_{pause} = 1.4$, empirically less than an audibly "bad" music transition; the intrapause penalty $\kappa_{intra} = .05$, cheaper than all but perfect music transitions; the music start and stop costs—which penalize otherwise free transitions—to small $\kappa_{enter} = \kappa_{exit} = .125$. Lastly, $\kappa_{changepoint} = 1$, which strongly favors inserting musical underlays whenever possible.

Score synthesis

The dynamic programming returns a sequence $\boldsymbol{s}$ of beat and pause indices. We then synthesize the final output score from $\boldsymbol{s}$ by concatenating the audio data for each beat and pause. We ensure smooth-sounding music transitions by inserting crossfades between beats s_i and s_{i+1} if they are not consecutive beats in the original music. In some cases, the musical score becomes out of alignment with the speech because music beats have variable length and we assume a fixed beat length on the speech. Our system corrects this problem every time the music pauses. If a pause segment ends at speech beat k, we start the music again exactly at speech beat k times the average beat duration regardless of the current time in the output. This re-aligns the score with the speech after every pause.

Our system adjusts the volume of the musical score so it is always audible but never overpowers the speech. We control the volume of the output score in two steps. First, we add volume curves for the score to follow: music fades in for three seconds after music pauses to a low level, increases exponentially to a high-level at speech emphasis points, decreases rapidly to a low level after music solos, and fades out for three seconds as music pauses begin. The second step adjusts these volumes relative to the speech volume. For every segment of the final output where both speech and music are playing, we adjust the decibel level of the music to lie 12 dB below the decibel level of the speech. Finally, we notch the 2.76 kHz and 5.63 kHz music frequencies by 6 dB because those frequencies contain important acoustic information for decoding consonants and vowels in speech [16].

RESULTS

We have generated 20 musical scores spanning seven music tracks and five speech tracks. We used all three of our emotion labeling techniques: hand-labeling, crowd-labeling, and automatic labeling. Table 1 shows our input speech tracks and summarizes the results we have generated, and Table 2 details our music tracks. Our supplemental material[1] includes all of these audio results. The first author of this paper labeled the emotions of each story by hand and we had an average of 17.5 crowd workers label each story at an average cost of $4.38 per story. The first author hand-labeled the music segments for all tracks, and we had an average of 18.5 crowd workers label the segments ($4.63 per track). Our algorithm computes the music track distance tables as a preprocess. On a 2.7 GHz Intel Core i7 processor with 8 GB of RAM, our system takes less than 1 second to find and generate the optimal musical score for a three minute story and two three minute music tracks. If we add music segment length constraints with d_{min} of 20 seconds and d_{max} of 90 seconds, the optimization takes 3 minutes. We generated all of our results with d_{min} of 20 seconds and d_{max} of 90 seconds except for "The Story of an Hour" musical score, which we use to demonstrate a longer story with a higher d_{min} (see supplemental material). We also include a basic example of a musical score that we generated for a children's book video.

[1] http://vis.berkeley.edu/papers/emotionscores

Figure 8. We visualize twelve of our generated results above. Each speech/music pair was generated using all three of our labeling methods. Evaluators on Mechanical Turk listened to three different scores for a speech track and the original speech without music. These charts show the average ranking in overall quality of the four clips among the evaluators.

Figure 8 shows a visualization of the twelve of our results—four speech/music combinations with each of the three labeling methods—which correspond to the bold-faced results in Table 1. The emotions in the speech and music consistently match in the hand- and crowd-labeled results. In two instances ("Roosts"/crowd and "Not That I Care"/crowd), an incorrect emotion plays in the music for a small number of beats, but neither is audibly noticeable. The automatic results do not match emotions as strongly. In "Not That I Care"/auto and "Goldilocks"/auto, the automatic approach matches large sections of "happy" speech with other music emotions. This happened because the automatic music labeling did not predict any "happy" segments of music, while the automatic speech labeling predicted spans of "happy" that are longer than the maximum length pause π_{max}.

Evaluation. To evaluate our results, we asked crowd workers to listen to and rank four versions of a story—*hand-labeled*, *crowd-labeled*, *automatically labeled*, and *no music*—in terms of overall quality. For each story, twelve US workers, each with over 95% approval rates on Mechanical Turk, evaluated the results. We rejected evaluation tasks if a worker did not spend enough time on the task to listen to all of the audio stories; we removed 24.6% of the ratings based on this test. Figure 8 shows the average ranking for each of the four versions for each speech track. Surprisingly, listeners ranked the results generated using crowd labels (average rank over the four stories of 2.21) as high as hand-labels (average of 2.3). Both methods generated higher quality results than automatic labels (average of 2.52) and the stories with no music (average of 2.95). We find the differences in rankings to be significant ($\chi^2(3) = 9.02$, $p < .05$) using Friedman's nonparameteric test for differences in ranks. Subsequent pairwise comparisons using the Wilcoxon signed rank test find a significant preference for crowd results over no music ($p < .005$) and hand-labeled results over no music ($p < .05$). This result suggests that the "wisdom of crowds" might apply to affective tasks. Our results also show that for all of the speech tracks, our system generates musical scores that improve in overall quality versus the speech without music.

To evaluate the consistency of the emotion labels, we compute Fleiss' and Cohen's kappa values. The average Fleiss' kappa is 0.271 between workers for the speech labels and 0.2 for the music labels. The average Cohen's kappa between our hand-labeled emotions and the most probable crowd labeling is 0.189 for speech and 0.437 for music, which suggests a large difference between the typical crowd-labeling and our expert hand-labeling. Finally, the reliability between our hand-labels and the automatic labels is nearly zero, at kappa of 0.0484 for speech and 0.136 for music. This discrepancy between human and automatic labels is a likely cause of the automatic method's low ranking in our listening evaluation.

CONCLUSION AND FUTURE WORK

Emotionally relevant musical scores can improve the quality and engagement of an audio story, but they are challenging and time-consuming to produce. We have presented a system for collecting emotion labels on speech and music, and an algorithm for generating an optimal score that matches emotions of music to speech. With this system, novice producers can generate high-quality musical scores for entire stories without needing audio editing expertise.

We believe there are promising directions in automatically editing multimedia based on emotion. However, applying our general method to scoring, for example, video and slideshows raises new challenges. An algorithm to generate musical scores for a visual medium must be able to insert pauses in the visuals to enable techniques like musical underlays. One approach may be to add video textures [26] that match the start and end frames in these pauses, as in Berthouzoz et al.'s [3] work on editing interview videos.

One bottleneck in applying our approach to more complex domains is in optimally setting the constraint parameters as we add new constraints to the optimization. We currently set these parameters by hand, but if the constraints become more complicated in the video and slideshow applications, we could apply Nonlinear Inverse Optimization (NIO) to learn a parameter assignment from example musical scores, following work from O'Donovan et al. [19] and Vollick et al. [30] in visual layout optimization.

Our system does not account for some characteristics of voice. In future work, we hope to study how the properties of voice, e.g., loud and rhythmic vs. soft and timid, impact the creation of relevant musical scores. Additionally, while our system uses emotion labels to match the music to speech, we plan to explore other labeling vocabularies that may provide new insight in linking music, speech, and visuals.

ACKNOWLEDGMENTS

This work is supported in part by a Google Faculty Research Award and the Intel Science and Technology Center for Visual Computing.

REFERENCES

1. Arikan, O., Forsyth, D. A., and O'Brien, J. F. Motion synthesis from annotations. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 402–408.
2. Audible. http://audible.com, Apr. 2014.
3. Berthouzoz, F., Li, W., and Agrawala, M. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 67.
4. Calvo, R. A., and D'Mello, S. Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on 1*, 1 (2010), 18–37.
5. CastingWords. https://castingwords.com, Apr. 2014.
6. El Ayadi, M., Kamel, M. S., and Karray, F. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition 44*, 3 (2011), 572–587.
7. Ellis, D. P., and Poliner, G. E. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, IEEE (2007), IV–1429–IV–1432.
8. Foote, J. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1, IEEE (2000), 452–455.
9. Foote, J., Cooper, M., and Girgensohn, A. Creating music videos using automatic media analysis. In *Proceedings of the tenth ACM international conference on Multimedia*, ACM (2002), 553–560.
10. Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H., and Cai, L.-H. Music type classification by spectral contrast feature. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, vol. 1, IEEE (2002), 113–116.
11. Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, B. G., Richardson, P., Scott, J., Speck, J. A., and Turnbull, D. Music emotion recognition: A state of the art review. In *Proc. ISMIR* (2010), 255–266.
12. Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 23–32.
13. Librivox. https://librivox.org, Apr. 2014.
14. Logan, B. Mel frequency cepstral coefficients for music modeling. In *ISMIR* (2000).
15. McFee, B., and Ellis, D. P. Learning to segment songs with ordinal linear discriminant analysis. In *Acoustics, Speech, and Signal Processing, 2014. ICASSP'14. Proceedings. 2014 IEEE International Conference on*, IEEE (2014).
16. McKee, J. Using music: The kitchen sisters. http://transom.org/2014/using-music-the-kitchen-sisters/, Feb. 2014.

17. Monteith, K., Francisco, V., Martinez, T., Gervas, P., and Ventura, D. Automatic generation of emotionally-targeted soundtracks. In *Proceedings of the Second International Conference on Computational Creativity* (2011), 60–62.

18. Monteith, K., Martinez, T. R., and Ventura, D. Automatic generation of music for inducing emotive response. In *Proceedings of the First International Conference on Computational Creativity* (2010), 140–149.

19. O'Donovan, P., Agarwala, A., and Hertzmann, A. Learning layouts for single-page graphic designs.

20. Posner, J., Russell, J. A., and Peterson, B. S. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology 17*, 03 (2005), 715–734.

21. Rubin, S., Berthouzoz, F., Mysore, G., Li, W., and Agrawala, M. Underscore: musical underlays for audio stories. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 359–366.

22. Rubin, S., Berthouzoz, F., Mysore, G. J., Li, W., and Agrawala, M. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, ACM (2013), 113–122.

23. Russell, J. A. A circumplex model of affect. *Journal of personality and social psychology 39*, 6 (1980), 1161.

24. Schmidt, E. M., and Kim, Y. E. Prediction of time-varying musical mood distributions from audio. In *ISMIR* (2010), 465–470.

25. Schmidt, E. M., and Kim, Y. E. Modeling musical emotion dynamics with conditional random fields. In *ISMIR* (2011), 777–782.

26. Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co. (2000), 489–498.

27. Schwarz, D. The caterpillar system for data-driven concatenative sound synthesis. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)* (2003), 135–140.

28. Schwarz, D., et al. A system for data-driven concatenative sound synthesis. In *Digital Audio Effects (DAFx)* (2000), 97–102.

29. Tauscher, J.-P., Wenger, S., and Magnor, M. A. Audio resynthesis on the dancefloor: A music structural approach. In *VMV*, M. M. Bronstein, J. Favre, and K. Hormann, Eds., Eurographics Association (2013), 41–48.

30. Vollick, I., Vogel, D., Agrawala, M., and Hertzmann, A. Specifying label layout style by example. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM (2007), 221–230.

31. Warriner, A. B., Kuperman, V., and Brysbaert, M. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods 45*, 4 (2013), 1191–1207.

32. Webster, T. The podcast consumer 2012. http://www.edisonresearch.com/home/archives/2012/05/the-podcast-consumer-2012.php, May 2012.

33. Wenger, S., and Magnor, M. A genetic algorithm for audio retargeting. In *Proceedings of the 20th ACM international conference on Multimedia*, ACM (2012), 705–708.

34. Wenner, S., Bazin, J.-C., Sorkine-Hornung, A., Kim, C., and Gross, M. Scalable music: Automatic music retargeting and synthesis. In *Computer Graphics Forum*, vol. 32, Wiley Online Library (2013), 345–354.

35. Willett, W., Ginosar, S., Steinitz, A., Hartmann, B., and Agrawala, M. Identifying redundancy and exposing provenance in crowdsourced data analysis. *Visualization and Computer Graphics, IEEE Transactions on 19*, 12 (2013), 2198–2206.

36. Yuan, J., and Liberman, M. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America 123*, 5 (2008), 3878.

37. Zickuhr, K., and Rainie, L. A snapshot of reading in america in 2013. http://www.pewinternet.org/2014/01/16/a-snapshot-of-reading-in-america-in-2013/, Jan. 2014.

APPENDIX - EFFICIENT OPTIMIZATION

Because our m and n are small (roughly $n = 500$ beats in a song, and roughly $m = 1000$ beats in the desired output), the entire table can fit in memory and we can use a dynamic programming algorithm to efficiently find this sequence of beats. However, music segment length constraints increase the size of this search space.

If we apply music segment length constraints, our goal is to find the lowest cost sequence of (beat, segment-length) pairs $\boldsymbol{s'}$, using T' and M' in place of T and M in Equation 1. The search space at each step has grown from size $(n + \pi_{max})^m$ to size $(n\delta_{max} + \pi_{max})^m$ because we need to consider the state space of all (beat, segment-length) pairs. The running time of a normal dynamic programming algorithm increases quadratically with δ_{max}. However, most of the entries in T' are infinity and are never in an optimal sequence of beats.

We take advantage of these hard constraints to implement an efficient version of this dynamic programming optimization whose runtime increases linearly with δ_{max}. We never explicitly compute T', which can grow prohibitively large. We define our new table T' in terms of blocks from T (see Figure 6), so we only store T. We then take advantage of the block structure of T' to limit our search space. For example, if the previous (beat, segment-length) pair in a path is (b_i, d_j), and segment length d_j is less than δ_{min}, our algorithm only needs to minimize over (beat, segment-length) pairs at length d_{j+1} instead of all possible (beat, segment-length) pairs.

Likewise, we do not construct the full table M'. Instead, we have a condition in our algorithm to check if speech bear $t = 1$, which forces the output beat/length pair to have length 1. In other cases, we tile δ_{max} copies of a column in M when our algorithm requires a column of M' during the optimization.