IDENTIFYING LOOPING COMPONENTS IN AUDIO MIXTURES

Sean Coffin

1. INTRODUCTION

The problem of identifying how many sources are present in a dense audio mixture and separating the contributions of these multiple sources remains a difficult problem with no known solution. While methods based on non-negative matrix factorization of magnitude spectrograms have experienced some success in these areas with application to general audio mixtures, such methods are still unable to identify the number of sources present in a mixture and do not yield ideal separation results, often giving separated components with highly objectionable artifacts. In hopes of developing new algorithms that may be successful in these difficult tasks we presently restrict our attention to a limited class of audio mixtures (simple mixtures consisting of looping components) and propose an unsupervised method for identifying both the number of components present over the duration of the mixture and when each of these components are contributing to the mixture, resulting in a transcription of the mixture into its component loops. While this restricted class of mixtures is a fair approximation of many modern pop songs which are based on the use of digital "loops" and "samples", thus making the resulting method fairly applicable to these mixtures directly, it is our hope that developing algorithms which are successful in this simpler, restricted class of mixtures will inform ways to approach the problem of source identification in the more complex case of general audio mixtures.

2. METHOD

2.1 Assumptions

As mentioned previously, the problem of source identification and separation of dense audio mixtures remains difficult, and thus we will restrict our attention to a class of mixtures with a high degree of time structure. Informally, we make the following assumptions about the mixtures at hand:

1. The audio mixture in question is comprised of different combinations of components selected from a small number of exactly repeating components, or "loops", and that these combinations change only at a particular set of times (which correspond to musical divisions of the song such as measures or beats).

2. The song maintains a tempo which is constant to the sample level and is given or has been found previously by some method.

3. The lengths of all "loops" being used to make the mixture are the same.

2.2 Method Overview

The proposed method is an unsupervised learning algorithm that consists of two parts: a higher level, general method for identifying the structure of an audio mixture in the class defined by the above assumptions using some feature space or representation of the audio mixture, and then lower level details regarding how the general method is adapted for a given feature space. In my work on this project I adapted the general method to implementations using both time-frequency (short time Fourier transform (STFT)) and stabilized auditory image (SAI) representations. While each of these representations afford different advantages and disadvantages in the task of component identification, I will focus on the application of the general method to STFT representations of audio mixtures in this report. The SAI representations were taught in another course this quarter (Psych 303 – Machine Hearing, taught by Richard Lyon), and my work with those representations will be neglected here in order to keep this report below the maximum specified length. The STFT method yielded better results than the SAI method in the implementations developed for this project, but the SAI method remains interesting as it would likely generalize well to audio mixtures that do not have exactly repeating components while the STFT method described here would not.

2.2.1 General Method

While the details of the general method may change slightly for various feature spaces, the individual steps taken during the course of the algorithm are guided by consistent overall principles. These principles, giving the general flow of the algorithm, will be described in this section.

The goal of the proposed method is to identify the number of components present over the duration of the mixture and when each of these components contributes to the mixture. Thus, the algorithm aims to build up a minimal "dictionary" of components that "explain" the mixture well; that is, a set of features (in

whatever feature space is being used) which can combine in some way to account for the full set of features given by the mixture. By attempting to keep this dictionary to a minimal number of elements and aiming to divide the feature space representation of the mixture in a way that captures its structure effectively, we hope to extract only dictionary elements that correspond to the components used to generate the audio mixture. The proposed method works toward this goal through an iterative algorithm that consists of two main steps:

1. Attempt to explain each mixture block with the current dictionary and intelligently use a "repeating / varying component decomposition" based method to either add new entries to the dictionary or modify the existing entries as needed to be able to explain all of the mixture blocks "well" (to within some error tolerance).

2. Prune the dictionary assembled in step 1 by removing redundant entries, entries that do not meet some minimal relevance criteria (that explain very little or nothing about the feature space of a mixture), or which have become fragmented in such a way that they are no longer plausibly part of any mixture block. If possible, improve the representations stored in each dictionary entry based on the complete set of mixture blocks in which they are plausibly components.

These two steps are repeated iteratively until the number of dictionary entries and the mixture blocks in which they are plausible components stabilize and remain unchanged from one iteration to the next.

2.2.2 Required Operations for Implementation in a Feature Space

Implementing the details of this general method in a particular feature space requires that we derive a way to implement two main operations in that space: a "repeating / varying component decomposition" of multiple mixture blocks' representations, and a way of determining if a dictionary entry is plausibly a component of a mixture block.

In order to form the dictionary elements, we will need some way of fragmenting the feature space representations of the mixture blocks into multiple components. This will be done by performing some kind of "repeating / varying component decomposition" in the feature space. What this means is that we will derive a method by which we can examine multiple blocks of the mixture in the given feature space and identify the portion of the feature space representation that remains the same across the blocks and the portion of the representation which varies, or differs, across the blocks. This is one way of letting the content of the mixture blocks themselves dictate how we should divide their feature spaces into relevant components, and is derived from the thought that when some portion of a mixture changes between two blocks while the rest remains the same we can conclude that these two portions of the mixture are able to change independently and thus come from two different components. By only splitting the feature space into components where the content of the mixture blocks is observed to change independently, we aim to derive a minimal set of components for describing that particular mixture without any pre-knowledge of the form of the individual components' representations in our feature space. Additionally, because we have assumed that the mixture contains only exactly repeating component waveforms ("loops"), this minimal set of components should be able to represent the mixture well without the components becoming overly fragmented (since we don't need to accommodate slight variations of recurring components).

In addition to forming dictionary elements, we will need to be able identify when a dictionary element is plausibly a component of a mixture block. Because our goal is to be able to explain our full mixture well with whatever dictionary we extract, we will need to be able to find if we have some combination of dictionary entries which explains each mixture block. Due to the fact that we may not always be working in a feature space where components of the mixture combine additively to produce the mixture's feature space representation (SAIs are the output of a non-linear, time-varying system, for example), we need a way of determining if a particular dictionary entry may plausibly be a component of a mixture block such that we can mark some corresponding portion of the mixture's representation as having been explained by that dictionary entry. This allows us to check for combinations of components that explain a mixture in a generalized way. In feature spaces where components combine linearly to produce the mixture, we may be able to implement simpler methods based on reconstruction error minimization.

The details of how these operations are used to implement the general method for the STFT feature space are presented in a flow chart on the next page. The notation used in this diagram will be explained in the next section.

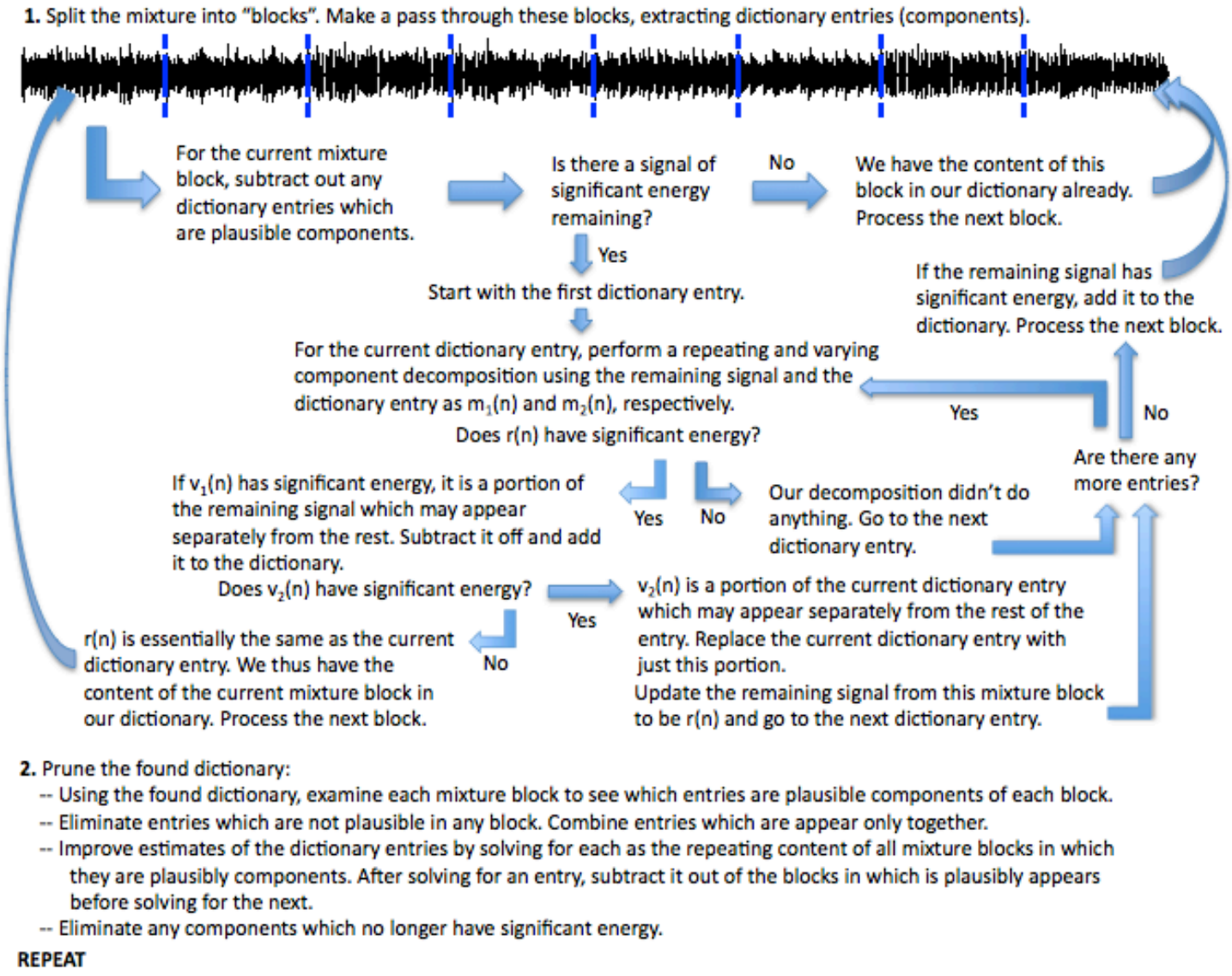

Fig 1. Flow chart showing the details of the general algorithm applied to the STFT feature space.

2.3 STFT Feature Space Specifics

As mentioned in the previous section, in order to use the overall method just described with a particular feature space or representation of an audio mixture, we must be able to both perform a "repeating / varying" component decomposition in that feature space and identify when components found using that decomposition are "plausible" components of a given mixture block. We will now describe the methods we have developed for implementing these two functions using an STFT representation of an audio mixture and follow up with a bit of commentary regarding details of the methods used and possible future improvements.

If we have multiple blocks of discrete time audio of the same length that we suspect have some common, repeatedly present component, we can denote these blocks in the time domain as

$$m_i(n) = r(n) + v_i(n) \quad i = 1, \ldots, k \quad (1)$$

where $m_i(n)$ denotes the mixture blocks, $r(n)$ denotes the component that is repeatedly present in all blocks, and $v_i(n)$ denotes the varying components in each block. If the repeating component and varying components are uncorrelated in each block, then the squared magnitude Fourier transforms of these signals (known as the PSD when considering statistical signals) combine additively and so we have

$$|M_i(\omega)|^2 = |R(\omega)|^2 + |V_i(\omega)|^2 \quad i = 1, \cdots, k \quad (2)$$

where $M_i(\omega)$, $R(\omega)$, and $V_i(\omega)$ denote the Fourier transforms of the mixture blocks, repeating component and varying components respectively. This gives that

$$|M_i(\omega)| \geq |R(\omega)| \quad i = 1, \ldots, k \quad (3)$$

and thus we can solve for the repeating component in a group of mixture blocks by forming magnitude spectrograms of those blocks, examining the magnitude values for each time-frequency bin across all mixture blocks, and taking the instance with the minimum magnitude to be the one in which there is the smallest contribution from a varying component and thus the best estimate of the repeatedly present component. We then take the complex values for each time-frequency bin from these "best estimate" bins and use them to reconstruct the time-domain repeating component. We can subtract this component from each mixture block to decompose them into a repeating and varying component, where the repeating component is present in all mixture blocks being considered. This procedure will theoretically recover the components exactly if the uncorrelated assumption holds.

The previous interpretation was given because it is the most natural interpretation of this repeating / varying decomposition and because equation (3) also informs how we check for component plausibility in this feature space: the mixture's magnitude spectrogram values should be greater than or equal to any plausible component's magnitude spectrogram values in all time-frequency bin, granted the uncorrelated assumption holds. Alternatively, we can view this decomposition as an application of ICA. If we take multiple mixture blocks that we again suspect contain some common, repeating content, we can take these blocks and consider them to be multiple observations of the same event. Additionally, if we assume equation (1) accurately describes the blocks, we expect that these k observations were generated by k+1 sources (r(n) and $v_i(n)$, i=1, …, k). We can then use the uncorrelated assumption stated above to formulate an underdetermined ICA problem, x = As, as

$$\begin{bmatrix} |M_1(\omega)|^2 \\ |M_2(\omega)|^2 \\ \vdots \\ |M_k(\omega)|^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ 1 & & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} |R(\omega)|^2 \\ |V_1(\omega)|^2 \\ |V_2(\omega)|^2 \\ \vdots \\ |V_2(\omega)|^2 \end{bmatrix} \quad (4)$$

where the mixing matrix A is k x k+1 dimensional and is known by our construction of the k "observations". We then solve for the k+1 components of the s vector by pursuing a sparse representation of the output using an l-1 norm minimization on the s vector while requiring that the x = As equality hold exactly and that the s vector contain only non-negative elements. This will result in the same magnitude assignments as the previously described method but recovers no information regarding phase. Phase can be recovered in a way similar to the previously described method.

Although the details of the implementation using this feature space are presented in Fig. 1, there are a few issues that are not shown in that figure. Because we subtract found components from the mixture at various points in the algorithm (when trying to explain mixture blocks with dictionary elements and when solving for updated component estimates in step 2 of the algorithm), the algorithm is sensitive to the order in which we consider the input mixture blocks and dictionary elements. This subtracting of components allows us to uncover low energy components that would otherwise be difficult to identify, but the resulting sensitivity to order of processing is undesirable. In order to minimize this sensitivity and improve the algorithm's robustness, the orders of the input blocks and dictionary elements are shuffled after each iteration of the algorithm. Additionally, it is likely that there are better ways of solving for the dictionary entries in part 2 of the algorithm using some kind of iterative method, and it may be possible (and better) to check if the dictionary contains a combination of components that explain a mixture block well by trying all possible combinations of all plausible components rather than sequentially checking for and subtracting out plausible components in some random order (the number of components will likely be small enough to make this tractable and this may remove some dependence on the order of evaluation if handled properly).

3. Experiment And Results

In working on this project, I constructed many simple mixtures that each contained three instrument tracks (drums, bass guitar, and electric piano) and were eight "blocks" long. For each mixture, the three instruments were assigned two different loops that they could play, and thus the mixtures were generated so that all eight possible combinations of these 6 component loops were presented across the duration of the mixtures. For each trial, the input to the algorithm was one of these mixtures cut into its eight blocks and the output was a count of the number of components determined to be present across the mixture and a "presence table" indicating when each of the found components was determined to be present in the mixture.

I ran the algorithm on 10 different mixtures (comprised of different component loop options for the three tracks). Because the results of the algorithm depend on the order of processing (even though shuffling is used to decrease this sensitivity), the algorithm was run for 10 trials on each mixture to get a measure of average performance for each mixture.

Because the results for one mixture did not save properly, I have presented only results from 9 of the 10 mixtures here. These results are summarized in Table 1 and Fig. 2.

Overall, the algorithm performed fairly well on the 9 mixtures attempted. While it was only able to identify the number of components and presence table correctly on one of the 10 mixtures, it never strayed too far from the correct outcomes, managing to capture some number of relevant components in all cases. In all but one of the mixtures, the algorithm failed to converge on a stable set of dictionary entries and was instead terminated after 5 iterations of the algorithm on each trial (this was found empirically to be enough iterations for the algorithm to essentially stabilize, even if the exact number and form of dictionary entries continued to fluctuate slightly during the iterations). Thus, it seems necessary to develop ways of dealing with this failure to converge and reasonable to expect that the algorithm has the potential to perform much better if this issue can be resolved.

Additionally, rough versions of the "repeating / varying decomposition" were used in processing in order to speed up processing time. While I feel it is unlikely that using the highest quality decomposition (using a hop size of 1 sample in the spectrogram representation used to solve for the decomposition) would greatly affect the results of this algorithm, it is possible that it would improve the results or make them more stable, as the extracted dictionary entries would contain fewer artifacts. The average processing time for these trials was about 70 minutes, using a hop size of 10 samples in the decompositions. This would imply an average processing time of about 700 minutes for a single trial at maximum resolution, a processing length that was long enough to deter use during development of the algorithm.

4. Conclusion

This algorithm is early in its development, and as of writing this report I already have ideas for several changes that are likely to improve these results such as comparing the presence tables resulting from updated component estimates to those used to derive the components to ensure that the updated components capture the structure they are intended to and searching for combinations of dictionary entries which explain mixture blocks well rather than sequentially subtracting out plausible components in a shuffled, arbitrary order. I look forward to continuing to work on this project and feel these results, while preliminary in nature, reflect a fair level of success in a difficult task that has previously not been attempted in this setting.

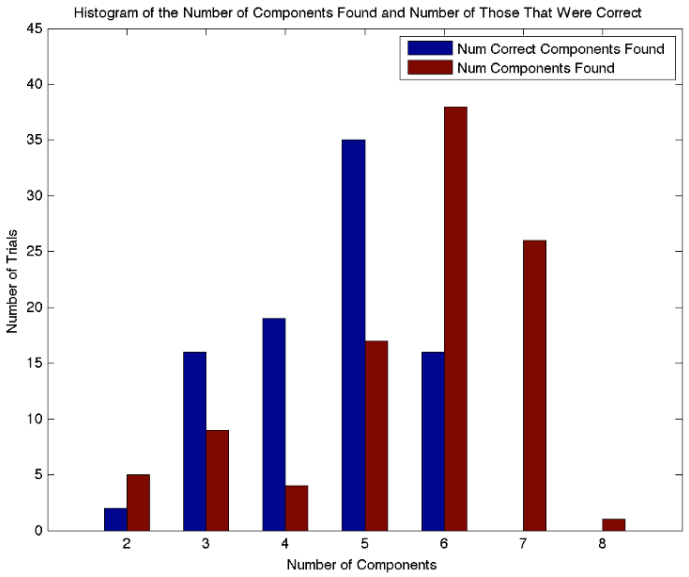

Fig. 2: Histogram of the number of components found by the algorithm in each trial and the number of these components that matched the presence pattern of a component used in generating the mixture (these are entirely accurately identified components). Note that the mixtures contained 6 different components and thus this is the maximum number of correctly identified components possible. The algorithm most often identified 6 different components and most often identified 5 components that matched actual components used in generation of the mixture

Mix	Mean Num	Std. Dev	Mean Correct	Std. Dev.	Mean % Correct	Std. Dev.
1	5.9	0.87	4.3	0.82	72.71	13.6
2	5.5	1.50	3.3	0.48	63.67	16.8
3	4.9	0.73	4.6	0.69	94.33	9.17
4	6.6	0.51	5.2	0.63	78.81	7.57
5	5.1	1.52	3.4	0.51	71.60	20.9
6	6.2	0.42	4.8	0.42	77.62	7.55
7	6.3	0.82	4.7	1.70	74.76	27.4
8	6.6	0.69	4.0	1.70	60.54	25.8
9	6.0	0.00	5.7	0.94	95.00	15.8
Tot	5.9	1.44	4.44	1.21	76.56	20.1

Table 1: Table of algorithm output statistics for the mixtures processed. "Mean Num" gives the mean number of components found by the algorithm for the mixture (over the 10 trials run), "Mean Correct" gives the mean number of these found components that had a presence pattern matching a true component used in construction of the mixture, and "Mean % Correct" gives the mean percent of found components that corresponded to correctly identified components. All "Std. Dev."s give the standard deviations associated with the immediately previous mean columns.