Collaborative Audio Enhancement: Crowdsourced Audio Recording

Minje Kim
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61874
minje@illinois.edu

Paris Smaragdis
University of Illinois at Urbana-Champaign
Adobe Research
paris@illinois.edu

Abstract

This paper presents a collaborative audio enhancement system that aims to recover a high-quality recording from multiple low-quality recordings provided by the crowd attending the same event. We see this procedure as a crowdsourcing example, because neither an automated system nor a set of crowdsourced recordings cannot easily replace a professionally processed manual audio recording, which is expensive and not always available. We do the job in the context where each recording is uniquely corrupted by different frequency responses of microphones, audio coding algorithms, interferences, noise, etc. To this end, we adopt a method of simultaneous probabilistic topic modeling on synchronized inputs, called Probabilistic Latent Component Sharing (PLCS). In PLCS, some of the parameters are fixed to be same during and after the learning process to capture the common audio content while the rest model unwanted recording-specific interferences and artifacts. The main contribution of the paper is that we speed up the EM-based PLCS parameter estimation process by incorporating Winner-Takes-All (WTA) hashing so that the update can be performed with fewer selective observations rather than the whole. Therefore, we can deal with a bigger set of recordings more efficiently. Experiments on music signals with various artifacts show that the proposed method provides with sensible speed-up with no degrade of audio quality compared with the comprehensive PLCS model.

1 Introduction

Because of widespread use of hand-held devices, we often find many overlapping recordings of an audio scene. Our goal in this paper is to efficiently utilize these low cost noisy data by extracting common audio sources from them so as to produce a higher quality rendering of the recorded event in a fast enough way. Hence, it can be seen as a collaborative approach to audio enhancement sharing some similar concepts with crowdsourcing methods [1, 2]. The first step towards unifying these recordings is to synchronize them, something we can easily achieve using one of the efficient and robust synchronization methods proposed in the past [3, 4]. Once this is done, one could simply use the best available recording at any point in time, assuming there is an automated way of quality-ranking the signals. This can be the simplest implementation of *collaborative audio enhancement*, where we can take advantage of other people's recordings to improve ours. However, such simple reasoning does not work for many common cases. Figure 1 shows a case where the obvious approach might fail. Between the two synchronized recordings, we cannot simply choose one because both are deficient, albeit in a different way. The bottom recording has a poor high frequency response, which could be the effect of a low-cost microphone or aggressive audio coding. On the other hand, the full bandwidth recording at the top has some interference in the 3 – 4.3 second region, which is however not present in the bottom one.

As the number of input recordings increases, the unique distortions in each recording make choosing a single best recording difficult, if not impossible. One could encounter various types of nonlinear artifacts or interferences, e.g., the audience chatter, lens zooming noises, button clicks, clipping, band-pass filtering, etc. Eventually we would like to solve this problem by using information from all recordings and combine it appropriately in order to produce a higher quality render.

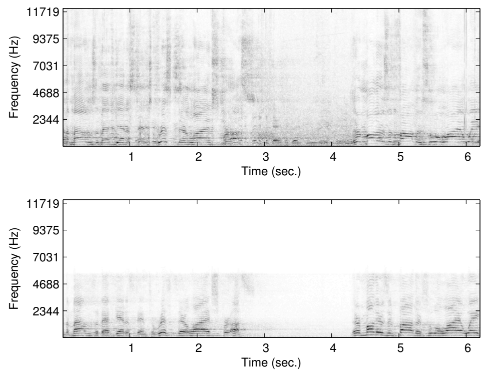

Figure 1: An example of a difficult scenario, when a synchronization and selection method can easily fail to produce a good recording. In this case we observe unwanted interference (top) and the other is band-limited (bottom).

Probabilistic Latent Component Sharing (PLCS) model [5] is based on the probabilistic counterparts of Nonnegative Matrix Factorization (NMF) [6, 7], such as Probabilistic Latent Semantic Indexing (PLSI) [8, 9]. PLCS extends PLSI with the common component sharing concept, similarly to Nonnegative Matrix Partial Co-Factorization (NMPCF) [10] does in the context of NMF. PLCS differs from the NMPCF-based methods in that it decomposes each input matrix into three parts, rather than just two, so that we can share both bases and encoding matrices while providing slack in the model by letting the weights of the components not to be shared. Because PLCS controls the contribution of the latent components with probabilistic weights, $P^{(l)}(z)$, it gives more intuitive interpretation of the roles of components in the reconstruction whereas in the Convolutive Common Nonnegative Matrix Factorization (CCNMF) model [11] they are absorbed in the filtering factor. Moreover, because the whole process is based on the probabilistic model, we could explicitly take advantage of Bayesian approaches, which is not straightforward in either NMPCF or CCNMF. The Bayesian approach provides a straightforward way to involve a certain amount of prior knowledge about the bases, which we can get in advance from the cleaner, but different versions of the similar sources.

There is room for improvement when we actually apply PLCS on the crowdsourced data. The EM-based estimation procedure of PLCS is not efficient enough to cope with big audio data from hundreds of recordings. In this paper we propose to harmonize a particular type of Locality Sensitive Hashing (LSH) [12], called Winner-Takes-All (WTA) hashing [13, 14], with the EM updates, so that each parameter is updated with a smaller set of relevant data samples rather than the whole. Since we rely on Hamming similarity between the hash codes of the parameter and the data sample, the construction of the relevant subset can be implemented in a cheap way by using bitwise operations.

2 Symmetric PLSI [9]

Given the magnitude spectrogram, $V = |X|$, with elements $V_{f,t}$ indexed by the frequency bin f and the time frame t, symmetric PLSI maximizes the log-likelihood $\mathcal{P}$ of observing the input $V_{f,t}$,

$$\mathcal{P} = \sum_{f,t} V_{f,t} \log P(f,t) = \sum_{f,t} V_{f,t} \log \sum_{z} P(f,t|z)P(z) = \sum_{f,t} V_{f,t} \log \sum_{z} P(f|z)P(t|z)P(z).$$

To get the second equality, the component-specific distributions $P(f,t,z)$ is further factorized into three parts: the frequency distribution $P(f|z)$, its temporal activations $P(t|z)$, and the component specific weights $P(z)$. Note that the term "symmetric" came from this tri-factorization [8], which eventually let us have control over additional temporal distributions of components as well as frequency distributions. This being a latent variable model, we use the Expectation-Maximization (EM) algorithm to estimate its parameters. In the E-step we find a posterior probability of the latent variable z given the time and frequency indices,

$$P(z|f,t) = \frac{P(f|z)P(t|z)P(z)}{\sum_z P(f|z)P(t|z)P(z)}. \quad (1)$$

In the M-step the expected complete data log-likelihood is maximized, which yields to the following update rules:

$$P(f|z) = \frac{\sum_t V_{f,t} P(z|f,t)}{\sum_{f,t} V_{f,t} P(z|f,t)}, P(t|z) = \frac{\sum_f V_{f,t} P(z|f,t)}{\sum_{f,t} V_{f,t} P(z|f,t)}, P(z) = \frac{\sum_{f,t} V_{f,t} P(z|f,t)}{\sum_{f,t,z} V_{f,t} P(z|f,t)}. \quad (2)$$

3 Probabilistic Latent Components Sharing [5]

Let us assume that there are L input magnitude spectrogram matrices, corresponding to L available recordings in the collaborative audio enhancement application. In PLCS we partition the values of the latent components in the l-th recording $z^{(l)}$ into two disjoint subsets, $z^{(l)} = z_C \cup z_I^{(l)}$, where z_C is the subset that contains indices of the common components shared across all recordings, and $z_I^{(l)}$ contains those of all the other components present only in the l-th recording. Now, the log-likelihood $\mathcal{P}$ of observing L given recordings can be written as:

$$\mathcal{P} = \sum_l \sum_{f,t} V_{f,t}^{(l)} \log \left\{ \sum_{z \in z_C} P_C(f|z) P_C(t|z) P^{(l)}(z) + \sum_{z \in z_I^{(l)}} P_I^{(l)}(f|z) P_I^{(l)}(t|z) P^{(l)}(z) \right\}. \quad (3)$$

The main feature in (3) is to fix both the spectral and the temporal distributions to be same across all inputs for $z \in z_C$, which are specified as the common variables $P_C(f|z)$ and $P_C(t|z)$. On the other hand, components indicated by $z \in z_I^{(l)}$ represent recording-specific sound components, such as interferences, characterized by parameters $P_I^{(l)}(f|z)$ and $P_I^{(l)}(t|z)$. We refer to this model as PLCS, for which the E-step is:

$$P^{(l)}(z|f,t) = \frac{P^{(l)}(f|z) P^{(l)}(t|z) P^{(l)}(z)}{\sum_{z \in z^{(l)}} P^{(l)}(f|z) P^{(l)}(t|z) P^{(l)}(z)}, \forall z \in z^{(l)}. \quad (4)$$

Note that the parameters $P^{(l)}(f|z)$ and $P^{(l)}(t|z)$ can either refer to the common parameters $P_C(f|z)$ and $P_C(t|z)$ when $z \in z_C$ or $P_I^{(l)}(f|z)$ and $P_I^{(l)}(t|z)$ when $z \in z_I^{(l)}$, respectively.

Using Lagrange multipliers to ensure that the probability distributions sum to one, we maximize the expected complete data log-likelihood with following update rules as the M-step:

$$\text{For } z \in z_I^{(l)}: \quad P_I^{(l)}(f|z) = \frac{\sum_t V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \quad P_I^{(l)}(t|z) = \frac{\sum_f V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \quad (5)$$

$$\text{For } z \in z_C: \quad P_C(f|z) = \frac{\sum_{l,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{l,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \quad P_C(t|z) = \frac{\sum_{l,f} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{l,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \quad (6)$$

$$\text{For } z \in z^{(l)}: \quad P^{(l)}(z) = \frac{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{z,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}. \quad (7)$$

Note that the updates for $P_C(f|z)$ and $P_C(t|z)$ include summation over l to involve all the reconstructions of common components.

3.1 Incorporating priors

It is often useful to involve prior knowledge about the parameters in probabilistic models. For instance, we can have a clean recording of the same content as in the provided inputs, albeit recorded at a different time (e.g. a studio recording of a song whose recordings we obtain from a concert). Or, it is also possible to assume that the interferences are a certain kind of sources, e.g. human voice. On the other hand, we cannot simply learn the bases of those a priori signals and fix them as our target parameters, $P_C(f|z)$ or $P_I^{(l)}(f|z)$, as there is no guarantee that the a priori known signals have exactly the same spectral characteristics with the target sources. To address this problem PLCS follows a Bayesian approach to derive a maximum a posteriori (MAP) estimator of the parameters.

First, we learn the bases of the magnitude spectrograms of the similar sources and interferences by directly applying PLSI update rules in (2). The learned bases vectors $P_{\text{source}}(f|z)$ and $P^{(l)}_{\text{interf}}(f|z)$ are used in the PLCS model to construct a new expected complete data log-likelihood

$$\begin{aligned}\langle\mathcal{P}\rangle = \sum_{l,f,t} V^{(l)}_{f,t}\Big\{ &\sum_{z\in z_C}\Big(P^{(l)}(z|f,t)\log P_C(f|z)P_C(t|z)P^{(l)}(z)\Big)\\ &+\sum_{z\in z_I^{(l)}}\Big(P^{(l)}(z|f,t)\log P_I^{(l)}(f|z)P_I^{(l)}(t|z)P^{(l)}(z)\Big)\Big\}\\ &+\alpha\sum_{f,z\in z_C}P_{\text{source}}(f|z)\log P_C(f|z)+\beta\sum_{l,f,z\in z_I^{(l)}}P^{(l)}_{\text{interf}}(f|z)\log P_I^{(l)}(f|z),\end{aligned}$$

where α and β controls the amount of influence of the prior bases. Once again, by using proper Lagrange multipliers, we can derive the final M-step with priors as follow:

$$\text{For } z\in z_I^{(l)}:\quad P_I^{(l)}(f|z)=\frac{\sum_t V^{(l)}_{f,t}P^{(l)}(z|f,t)+\beta P^{(l)}_{\text{interf}}(f|z)}{\sum_{f,t}V^{(l)}_{f,t}P^{(l)}(z|f,t)+\beta P^{(l)}_{\text{interf}}(f|z)}, \tag{8}$$

$$\text{For } z\in z_C:\quad P_C(f|z)=\frac{\sum_{l,t} V^{(l)}_{f,t}P^{(l)}(z|f,t)+\alpha P_{\text{source}}(f|z)}{\sum_{l,f,t}V^{(l)}_{f,t}P^{(l)}(z|f,t)+\alpha P_{\text{source}}(f|z)}. \tag{9}$$

E-step and the other M-step update rules are not changed from the original PLCS model. Figure 2 summarizes the whole PLCS process on three different inputs: low-pass filtered, high-pass filtered, and mid-pass filtered inputs. All three inputs also contain unique distortions represented with different noise patterns in the figure.

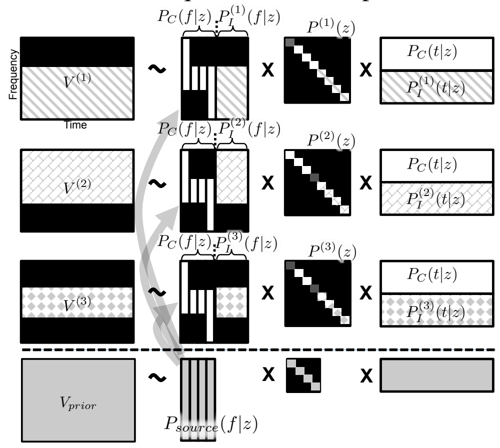

Figure 2: An example of common source separation process using PLCS on three defected input matrices and prior information.

Note that the first common component of $l = 1$ case (first row) degrades the reconstruction as its basis vector has high frequency energy while $V^{(1)}$ was low-pass filtered. Therefore, the first weight in the diagonal matrix $P^{(1)}(z = 1)$ has a very low (dark) value. Similarly, $P^{(2)}(z = 4)$, $P^{(3)}(z = 1)$, and $P^{(3)}(z = 4)$ are also those weights that *turn off* inactive common components. Note also that the a priori learned bases $P_{\text{source}}(f|z)$ are full-banded and have somewhat different spectral shapes from the common bases, so they cannot replace the common bases as they are.

To recover the magnitudes of the desired sources, we multiply the sum of the posterior probabilities of $z \in z_C$ to the input complex-valued spectrograms $X_{f,t}$,

$$\hat{S}^{(l)}_{f,t}=X^{(l)}_{f,t}\sum_{z\in z_C}P^{(l)}(z|f,t),$$

where $\hat{S}^{(l)}$ is the spectrogram of the separated sources from the l-th input.

3.2 Post Processing

It is possible that the recorded signals exhibit non-uniform frequency responses due to recording devices and format specifications. The PLCS method can identify the isolated common sources, but it is not expected to ameliorate effects like frequency response losses, since that information will be coded in the basis vectors and is not readily accessible as an artifact. A collaborative post processing step addresses this issue. It is motivated by the fact that even if most of the recordings are filtered in some way, one recording that did not go through such filtering can give us enough

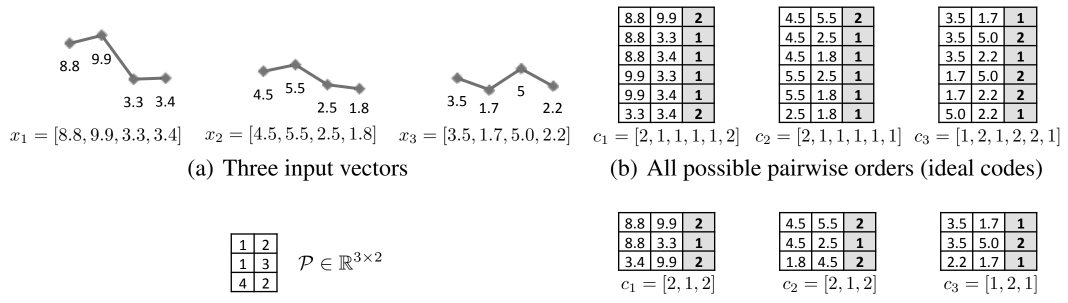

Figure 3: A WTA hashing example. (a) x_1 and x_2 are similar in their shape while x_3 looks different. (b) Exhaustive pairwise ordering can give ideal hash codes c, but the feature space (hash code) is high dimensional. (c) A permutation table provides a fixed set of random pairwise selections. (d) $\mathcal{P}$ generates succinct hash codes that approximate the original similarity of inputs.

information to recover the full-banded reconstruction. Suppose that we get L recovered spectrograms $\hat{S}^{(l)}_{f,t}$ using PLCS. The post processing begins with getting the normalized average magnitude spectrum of those reconstructions $y^{(l)}_f = \sum_t |\hat{S}^{(l)}_{f,t}| \Big/ \sum_{f,t} |\hat{S}^{(l)}_{f,t}|$. Now, we can obtain some global weights by considering the balance among different recordings in each particular frequency bin, $w_f = \sum_l y^{(l)}_f \Big/ \max_l y^{(l)}_f$. Then, the final complex spectrogram of the desired output is obtained by dividing the sum of the band-limited reconstructions with the corresponding elements of the global weight: $\hat{S}_{f,t} = \frac{\sum_l \hat{S}^{(l)}_{f,t}}{w_f}$.

3.3 Computational complexity

If each l-th input is a matrix of size $F \times T$, and we assume that there are $Z = Z_C + Z_I$ latent components, where Z_C and Z_I respectively stand for the number of common and individual ones, calculation of posterior probabilities (4) runs in the order of $\mathcal{O}(FTZL)$. In the M-step (5), (6), and (7), each update requires another set of operations in the order of $\mathcal{O}(FTZL)$. When we have many recordings to deal with (big L) or the length of each recording is long (big T), the computational complexity grows proportionally to them. In this paper we propose to use hashing techniques to reduce the size of F and T.

4 Winner-Takes-All Hashing

The recent application of WTA hashing [13] to a big image searching task provided accurate and fast detection results [14]. As a kind of locality sensitive hashing [12], it has several unique properties: (a) similar data points tend to collide more (b) Hamming distance of hash codes approximately reflects the original distance of data. Therefore, it can be seen as a distribution on a family of hash functions $\mathcal{F}$ that takes a collection of objects, such that for two objects x and y, $\Pr_{h\in\mathcal{F}}[h(x) = h(y)] = \text{sim}(x, y)$. $\text{sim}(x, y)$ is some similarity function defined on the collection of objects [15].

WTA hashing is based on the rank correlation measure that encodes relative ordering of elements. Although the relative rank order can work as a stable discriminative feature, it non-linearly maps data to an intractably high dimensional space. For example, the number of orders in M-combinations out of an F-dimensional vector is (# combinations) $\times$ (# orders in each combination) $= \frac{F!}{M!(F-M)!} \times M$. Instead, WTA hashing produces hash codes that compactly approximate the relationships.

WTA hashing first defines a permutation table $\mathcal{P} \in \mathbb{R}^{K\times M}$ that has K different random index sets, each of which chooses M elements. For the k-th set the position of the maximal element among M elements is encoded instead of the full ordering information. Therefore, the length of hash codes is MK-bits since each permutation results in M bits, where only one bit is on to indicate the position of the maximum, e.g. $3 = 0100$, and there are K such permutations. Whenever we do this encoding for an additional permutation, at most $M-1$ new pairwise orders (maximum versus the others) are embedded in the hash code. The permutation table is fixed and shared so that the hashing results are consistent. Figure 3 shows the hashing procedure on simple data. Note that the Euclidean distance between x_1 and x_2 are larger than that of x_2 and x_3 as opposed to the similarity in their shapes. WTA hashing results in hash codes that respect this shape similarity. Note also that WTA hashing is robust to the difference between x_1 and x_2, which can be explained as some additive noise.

5 The Proposed Speed-up: EM for PLCS on Nearest Neighbors

The M-step of PLCS estimation from (5) to (7) (or with (8) and (9) in the MAP version) can be seen as a weighted summation of the input matrices, where the weights are defined by the estimation of posterior probabilities in (4). We make this re-weighting procedure faster by focusing only on the ones with high posterior probabilities, but it requires the computation of posterior probabilities anyway. Instead, we construct a pseudo-nearest neighbor set based on the Hamming similarity between the parameter and input, hoping that it serves as good candidates. For example, for a basis vector with a fixed z, $P(\mathbf{f}|z)$, in the PLSI case, we can update it based on its neighbors, such as

$$P(\mathbf{f}|z) = \frac{\sum_t V_{f,t} P(z|f,t)}{\sum_{f,t} V_{f,t} P(z|f,t)} \approx \frac{\sum_{t\in\mathcal{N}_z^c} V_{f,t} P(z|f,t)}{\sum_f \sum_{t\in\mathcal{N}_z^c} V_{f,t} P(z|f,t)}, \tag{10}$$

where $\mathcal{N}_z^c$ denotes the set of column vectors of the input that are closest to the current estimation of the z-th basis vector, $P(\mathbf{f}|z)$, where the bold character $\mathbf{f}$ is introduced to denote all possible realization of random variable $\mathbf{f}$, i.e. z-th column vector of the matrix $P(\mathbf{f}|\mathbf{z})$. Similarly, we can define $\mathcal{N}_z^r$ for the set of row vectors.

This approximation with nearest neighbors intuitively makes sense as we can think of the nearest neighbors to the current parameters as *purer* data points where the contribution from the other components are less significant. A possible disadvantage would be the case if the current estimation is stuck in some isolated local minimum, which is surrounded by very low sample probabilistic densities. In this worst case scenario, nearest data samples that are all from the isolated reason are not able to help the estimation escape from the local minima.

We also have to be careful about choosing the nearest neighbors. An optimal divergence measure will be cross entropy in our probabilistic topic models, but then searching takes up another order of $\mathcal{O}(FTL)$ computations per each parameter vector. We instead adapt WTA hash codes as our features, so that the nearest neighbors can be defined based on Hamming similarity between them as follows:

$$\sum_k^{KM} h\big(V_{\mathbf{f},t\in\mathcal{N}_z^c}\big) \wedge h\big(P(\mathbf{f}|z)\big) > \sum_k^{KM} h\big(V_{\mathbf{f},t\notin\mathcal{N}_z^c}\big) \wedge h\big(P(\mathbf{f}|z)\big), \tag{11}$$

where the hash function $h(\cdot)$ takes an input column vector and convert it into a $KM \times 1$ dimensional binary vector. Hence, similar hash codes yield more agreements in their bits, which can be captured by bitwise AND ($\wedge$) operation followed by bitwise summation. The inequality holds if the set of neighbors $\mathcal{N}_z^c$ are the ones that are closest to the current estimation of $P(\mathbf{f}|z)$. We dropped the recording index l for the notational simplicity, but there are L sets of neighbors for $P(\mathbf{f}|z)$.

Finally, the update rules of PLCS can be simplified with the hashing concept:

$$\text{For } z \in z_I^{(l)}: \quad P_I^{(l)}(f|z) = \frac{\sum_{t\in\mathcal{N}_z^{c(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \beta P_{\text{interf}}^{(l)}(f|z)}{\sum_f \sum_{t\in\mathcal{N}_z^{c(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \beta P_{\text{interf}}^{(l)}(f|z)}, \tag{12}$$

$$P_I^{(l)}(t|z) = \frac{\sum_{f\in\mathcal{N}_z^{r(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{f\in\mathcal{N}_z^{r(l)}} \sum_t V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \tag{13}$$

$$\text{For } z \in z_C: \quad P_C(f|z) = \frac{\sum_l \sum_{t\in\mathcal{N}_z^{c(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \alpha P_{\text{source}}(f|z)}{\sum_{l,f} \sum_{t\in\mathcal{N}_z^{c(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \alpha P_{\text{source}}(f|z)}, \tag{14}$$

$$P_C(t|z) = \frac{\sum_l \sum_{f\in\mathcal{N}_z^{r(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{l,t}\sum_{f\in\mathcal{N}_z^{r(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \tag{15}$$

$$\text{For } z\in z^{(l)}: \quad P^{(l)}(z) = \frac{\sum_{f\in\mathcal{N}_z^{c(l)}} \sum_{t\in\mathcal{N}_z^{r(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_z \sum_{f\in\mathcal{N}_z^{c(l)}} \sum_{t\in\mathcal{N}_z^{r(l)}} V_{f,t}^{(l)} P^{(l)}(z|f,t)}. \tag{16}$$

Note that after each update, we rebuild the neighbor sets based on the fresh parameter estimates. Also, we properly add recording index l on the neighbor sets, i.e. $\mathcal{N}_z^{c(l)}$ and $\mathcal{N}_z^{r(l)}$.

5.1 Computational complexity

Update rules from (12) to (16) reduce the original complexity $\mathcal{O}(FTZL)$ down to $\mathcal{O}(FrTZL)$, $\mathcal{O}(rFTZL)$, and $\mathcal{O}(rFrTZL)$, for the vertical, horizontal, and diagonal parameter vectors, respectively, where the proportional size of the neighbor set r defines the actual degree of compression. For example, $r = 0.1$ picks up only 10% of the column or row vectors. As for the nearest neighbor search, we still have to perform $\mathcal{O}(ZFL)$ or $\mathcal{O}(ZTL)$ comparisons among the hash codes generated from all the input vectors versus the parameter vectors, but each comparison on KM binary values is with minimal cost since they can be implemented in a cheap way.

In practice, E-step can be performed more often rather than once after all the parameter updates in the M-step. For instance, one can update the posterior probabilities after every line of the M-step from (12) to (16). However, now we do not need the entire posterior probabilities, which need the order of $\mathcal{O}(FTZL)$ computations. Instead, we can instantly calculate the only relevant elements. For example, for the updates in (12) and (14), we can prepare some reduced posterior probabilities such as,

$$P^{(l)}(z|f,t') = \frac{P^{(l)}(f|z)P^{(l)}(t'|z)P^{(l)}(z)}{\sum_{z\in z^{(l)}} P^{(l)}(f|z)P^{(l)}(t'|z)P^{(l)}(z)}, \forall z\in z^{(l)} \text{and } \forall t'\in\mathcal{N}_z^{c(l)}, \tag{17}$$

because the other elements with $t \notin \mathcal{N}_z^{c(l)}$ are not used in the succeeding M-step. Likewise, $P^{(l)}(z|f',t)$ with $f' \in \mathcal{N}_z^{r(l)}$ for (13) and (15), and $P^{(l)}(z|f',t)$ with $f' \in \mathcal{N}_z^{r(l)}, t' \in \mathcal{N}_z^{c(l)}$ for (16) can be also defined with lower complexities.

6 Experimental Results

In this section, we compare the original PLCS model with the hashing-based approximation we propose in this paper. Their performances are compared in terms of signal-to-distortion ratio (SDR) [16], because we desire to reduce all the artifacts, noises, and interferences. Because the proposed method is less complex than the original PLCS model, the goal of this comparison is to see if the speed-up introduces any performance drop. To this end, we use five different single channel songs with 44.1kHz sampling rate and 16 bit encoding, each of which has a pair of versions: a 15 seconds-long clean live recording S as the source and a 30 seconds-long clean studio recording S_{prior} as the prior information of the source. The professional live recording S goes through three different sets of artificial deformations to simulate usual recording scenarios. The resulting three mixture spectrograms are:

- $x^{(1)}$: Low-pass filtering at 8kHz (a recording with a low sampling rate) / additional female speech as an interference
- $x^{(2)}$: High-pass filtering at 500Hz / additional female speech different from $x^{(1)}$ as an interference
- $x^{(3)}$: Low-pass filtering at 11.5kHz / high-pass filtering at 500Hz / clipping.

Short-time Fourier transform was applied to the signals with following settings: 1024 sample frame-length and 512 sample of hop size, and then we take the magnitude to construct our nonnegative input matrices $V^{(l)} = |X^{(l)}|$. For the priors, we get 100 bases for the source prior $P_{\text{source}}(f|z)$ from the studio recording S_{prior} while 50 interference prior bases $P_{\text{interf}}(f|z)$ are learned from anonymous female speeches [17].

We set up the two models as follows:

- The original PLCS model: This full PLCS model uses both the source and interference priors, $P_{\text{source}}(f|z)$ and $P_{\text{interf}}(f|z)$ to initialize them and learn them using both (8) and (9). We initialize $P_{\text{source}}(f|z)$ with a hundred bases learned from the studio recording, while $P_{\text{interf}}(f|z)$ by 50 bases learned from anonymous female speech. The initial construction was done using the ordinary PLSI algorithm. Note that we do not assume the interference prior for $X^{(3)}$, because it is riddled with clipping, not an additional interference.
- The proposed hashing-based model: We define two permutation matrices $\mathcal{P}^c, \mathcal{P}^r \in \mathbb{Z}_+^{128\times 2}$ for indexing elements in a column or a row, respectively. Therefore, they can hold indices up to F and T as its elements, respectively. After hashing, each column or row vector is transformed into a binary hash code vector with length 128×2, each of whose k-th pairs holds the maximum of the two compared elements indicated by k-th row of $\mathcal{P}^c$ or $\mathcal{P}^r$.

Figure 4 shows the average of SDR improvements from the runs of both systems on five different music signals. The most noticeable observation is that the hashing methods quickly reach the highest SDR values, but more iterations result in worse separation. This is a very common behavior of algorithms developed from NMF or PLSI, since the original log-likelihood error has nothing to do the separation quality, but it is about the overall approximation quality. With hashing, we observe much more drastic change of SDR values over iterations, which is a minus for the proposed system, because the performance depends more on the number of iterations. However, the actual separation performance itself is not worse than the comprehensive model. As expected, more number of neighbors provides more stable performance (thicker lines). As for the priors, we gently reduce the impact of prior information as the iteration i increases ($\alpha = \beta \propto e^{-i}$).

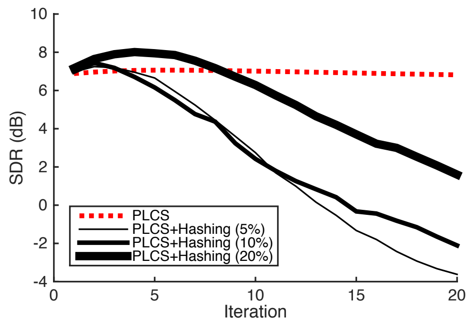

Figure 4: The audio enhancement quality of the systems in terms of SDR.

For now, our MATLAB® implementation of the proposed method is not actually faster than the full PLCS model, since it is difficult to beat the native matrix computations in MATLAB® with an implementation with loops. However, we believe that the merit of our proposed method is already clear with the computational complexity analysis, and more hardware-friendly implementations will exhibit the merit.

7 Conclusion

We extended the PLCS model for collaborative audio enhancement, by limiting the number of participating observations during the EM-based updates. Since the selection is based on the nearest neighbor search in terms of the Hamming similarity between hash codes, the cost of this additional search step is minimal. As a result, we dropped the complexity of the EM updates down to $r < 1$, which is a big advantage as the task requires the algorithm to be fast. On top of that, the proposed hashing-based extension inherits the original characteristics of the PLCS model, i.e. its ability to distinguish common audio sources from recording-specific artifacts by sharing some part of spectral and temporal marginal factors. The advantage of the proposed efficient model was shown by experiments on artificially contaminated music signals by providing comparable performance to the original PLCS model. We leave the more proper implementation and experiments on bigger datasets to future work.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1319708.

References

[1] D. C. Brabham, “Crowdsourcing as a model for problem solving: An introduction and cases,” *Convergence: The International Journal of Research into New Media Technologies*, vol. 14, no. 1, pp. 75–90, 2008.

[2] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, “Learning from crowds,” *Journal of Machine Learning Research*, vol. 11, pp. 1297–1322, Aug. 2010.

[3] N. J. Bryan, P. Smaragdis, and G. J. Mysore, “Clustering and synchronizing multicamera video via landmark cross-correlation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[4] P. Shrestha, M. Barbieri, H. Weda, and D. Sekulovski, “Synchronization of multiple camera videos using audio-visual features,” *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 79–92, 2010.

[5] M. Kim and P. Smaragdis, “Collaborative audio enhancement using probabilistic latent component sharing,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[6] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

[7] ——, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 13. MIT Press, 2001.

[8] T. Hofmann, “Probablistic latent semantic indexing,” in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[9] ——, “Probablistic latent semantic analysis,” in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

[10] M. Kim, J. Yoo, K. Kang, and S. Choi, “Nonnegative matrix partial co-factorization for spectral and temporal drum source separation,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1192–1204, 2011.

[11] P. Leveau, S. Maller, J. Burred, and X. Jaureguiberry, “Convolutive common audio signal extraction,” in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, 2011, pp. 165–168.

[12] P. Indyk and R. Motwani, “Approximate nearest neighbor – towards removing the curse of dimensionality,” in *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 604–613.

[13] J. Yagnik, D. Strelow, D. A. Ross, and R. Lin, “The power of comparative reasoning,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011, pp. 2431–2438.

[14] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, “Fast, accurate detection of 100,000 object classes on a single machine,” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[15] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, NY, USA, 2002.

[16] E. Vincent, C. Fevotte, and R. Gribonval, “Performance measurement in blind audio source separation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[17] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, “TIMIT acoustic-phonetic continuous speech corpus,” *Linguistic Data Consortium, Philadelphia*, 1993.