Signal Processing 111 (2015) 61–72

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Self-content-based audio inpainting

Yuval Bahat [a,*], Yoav Y. Schechner [a], Michael Elad [b]

[a] Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel
[b] Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 4 June 2014
Received in revised form
13 November 2014
Accepted 27 November 2014
Available online 5 December 2014

Keywords:
Packet loss concealment
Example-based inpainting

ABSTRACT

The popularity of voice over Internet protocol (VoIP) systems is continuously growing. Such systems depend on unreliable Internet communication, in which chunks of data often get lost during transmission. Various solutions to this problem were proposed, most of which are better suited to small rates of lost data. This work addresses this problem by filling in missing data using examples taken from prior recorded audio of the same user. Our approach also harnesses statistical priors and data inpainting smoothing techniques. The effectiveness of the proposed solution is demonstrated experimentally, even in large data-gaps, which cannot be handled by the standard packet loss concealment techniques.

1. Introduction

Voice over Internet protocol (VoIP) systems have become a basic tool with ever growing popularity. However, they commonly rely on an unreliable communication channel, such as the Internet, and are therefore subject to frequent events of data loss. These events are usually realized as lost data packets carrying audio information. This, in turn, leads to temporal gaps in the received audio sequences, as illustrated in Fig. 1. Left untreated, such gaps create breaks in the audio (e.g. missing syllables in speech signals). High percentage of packet loss (above 20%) can often render speech unintelligible [1]. For this reason, VoIP applications regularly incorporate a *packet loss concealment* (PLC) mechanism, to counter the degradation in audio quality, by filling in for the missing audio data, using various techniques.

A PLC mechanism should not impose high computational loads or extensive memory usage. Specifically, PLC should operate in real-time. Moreover, intense computations consume more power, which is a limited resource in mobile devices.

Most existing PLC techniques have difficulties handling long audio gaps. This paper presents an approach for handling such gaps, corresponding to high packet loss rates. We suggest using an example-based principle that exploits audio examples collected from past audio signals. Once an audio gap is encountered, our algorithm harnesses the audio data surrounding this gap to look for the most suitable audio example to fill this gap. A mixture of audio features and prior knowledge on the statistical nature of the audio signal is used for finding the most appropriate set of examples that could be used for filling the gap. Once found, our solution presents a series of steps for isolating the best fitted example to use and pre-processing the exact portion of the audio to be extracted from the chosen example. This portion is smoothly inlaid to fill the audio gap.

Inpainting is a term commonly used in the context of filling in missing pixels in images. It was borrowed by Adler et al. [2] to describe filling short audio gaps in a signal, by using the intact portions surrounding each gap. Our work has a similar flavour, but it differs from [2] in several important aspects. The novelty in our work lies in using a self-content-based approach, while exploiting a higher level model for the audio signal. These enable handling longer temporal audio gaps which [2] cannot handle, as observed when experimenting with such long gaps.

* Corresponding author.
E-mail addresses: yuval.bahat@gmail.com (Y. Bahat), yoav@ee.technion.ac.il (Y. Y. Schechner), elad@cs.technion.ac.il (M. Elad).

http://dx.doi.org/10.1016/j.sigpro.2014.11.023
0165-1684/

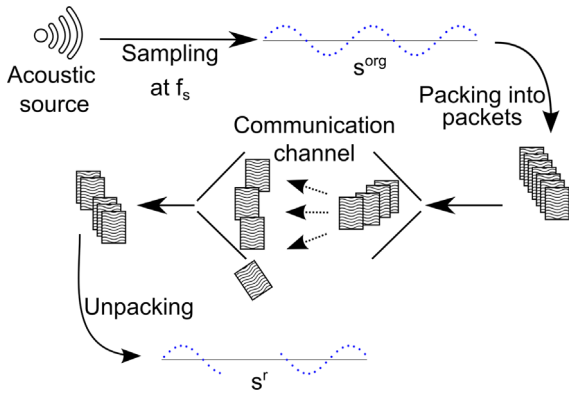

Fig. 1. Packet drop scenario. Some packets are dropped during transmission, causing $\mathbf{s}^r$ to have sequences of missing samples.

The rest of this paper is structured as follows. Section 2 explains the problem of loosing network packets, and surveys several existing methods to overcome this problem. Section 3 comprises a general high level description of the proposed method. We formulate the problem in Section 4. Then, Section 5 discusses the use of a statistical prior in a matching process, described thoroughly in Section 6. The consecutive process of inlaying the chosen example into the audio signal is described in Section 7. In Section 8 we analyze the method's computational complexity. Experimental results are described in Section 9 and discussed in Section 10. We conclude this paper in Section 11. For convenience, a table of notations is given at the end of the paper (See Table 4 in Appendix section).

2. Loosing network packets

The building block of VoIP is an Internet packet which encapsulates a segment of a digital audio signal. Let L^{packet} be the number of audio samples contained within each packet. Packets may have various sizes, corresponding to different values of L^{packet}. Ref. [1] mentions packets corresponding to 10, 20, 30 and 40 ms of audio. For a sampling rate of 8 KHz, these packets have $L^{\text{packet}} = 80, 160, 240$ and 320 samples, respectively. Packets frequently get dropped, often due to deliberate action in times of network congestion. This results in loss of the encapsulated data they carry.

In [1], Ding and Goubran showed the dramatic influence of lost packets (in various loss rates and packet sizes), and examined different PLC techniques. Some techniques are described in [3,4], and can be roughly divided into sender-based and receiver-based methods. Sender based methods (e.g. FEC) involve sending auxiliary information to allow later reconstruction. The auxiliary information maximizes redundancy while consuming minimal bandwidth. Such PLC methods require modifications in both sender and receiver.

Our proposed method only involves the receiving side, hence it is receiver-based. Receiver-based methods typically require lower bandwidth, or allow higher quality for a given bandwidth. There is a variety of receiver-based methods, some substitute a missing packet, either by a repetition of the adjacent preceding packet or by a predefined audio (noise or silence segment [5]). Other methods include waveform linear predictions [6,7] and codec-dependent spectral interpolation [8–10]. All of these are reported to perform adequately for short temporal losses (up to around 20 ms), but brake down for longer periods of time [1].

However, long gaps are common [11]. The Gilbert model for Internet packet loss [12] implies that packet dropping tends to occur in bursts, mainly when network congestion is experienced. This model fits packet loss statistics rather accurately. Using the model with standard parameters [11] suggests two important characteristics, which are taken into consideration in this work:

1. Dropping bursts of more than 5 consecutive packets are highly improbable, even in a poor quality communication channel.
2. When dealing with larger packet sizes (corresponding to longer encapsulated audio segments), gaps longer than 40 ms are highly probable.

3. Algorithm sketch

The PLC process starts by continuously capturing a streaming digital audio signal. This audio signal is divided on the fly into overlapping segments of constant length. We call these segments *audio blocks* (ABs). Audio blocks in our system are substantially longer than a packet, for reasons that will be clarified later on. Each AB undergoes a feature extraction process, which yields a feature vector representative of this AB.

During time periods where packets are not dropped, our system collects ABs and saves them as reference *example ABs* to be used at a later stage. Once a packet is dropped, the received audio has a missing sequence of samples. This missing sequence is a *hole* in all partially overlapping ABs that contain this sequence ($\mathbf{q}_n$ and $\mathbf{q}_{n+1}$ in Fig. 2).

ABs that contain the hole constitute a set of optional *query ABs*, which share the same length as example ABs. In queries only, the part of the query AB corresponding to the hole is blank. The unharmed portions within these queries undergo a feature extraction process, similar to the one applied to example ABs. This process yields query feature vectors, to be compared to example feature vectors.

The remainder of this section provides a cursory description of our algorithm (see Fig. 2). Readers seeking rigorous formulation are encouraged to skip directly to Section 4. For each query, we pick the most suitable example to fill the hole. This selected example is the one best satisfying a weighted combination of the following requirements:

1. *Low feature space distance*: This reflects the demand that for each hole, the intact portions of the query AB and its corresponding portions of the chosen example AB are similar.
2. *High prior probability for the resulting AB sequence*: We model the AB sequence as a hidden Markov chain. Then, the prior is the probability of the chosen example AB appearing between the ABs that proceed and succeed the query AB.

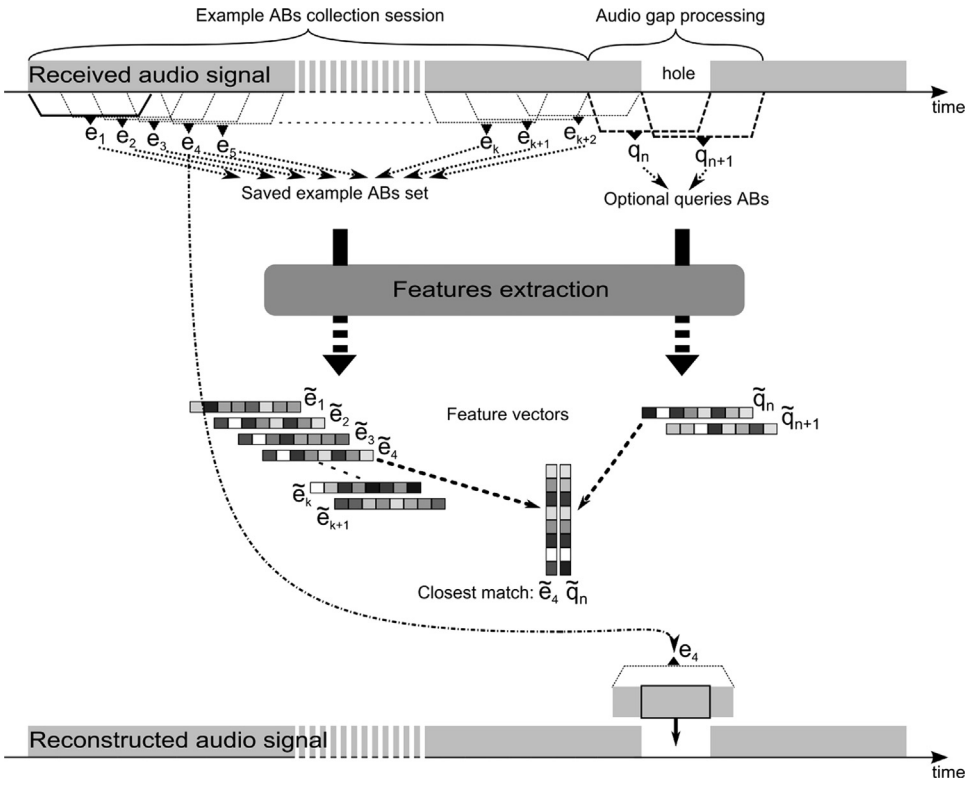

Fig. 2. Algorithm's sketch (Described in Section 3).

Finally, to reduce artifacts and increase intelligibility while inlaying the selected example, we conduct the following steps:

1. Pitch modification of the chosen example AB, when the hole is located within a voiced syllable. This modification is intended to prevent artifacts resulting from the difference of pitch between the query and example ABs. Such difference can occur when the same syllable is uttered in different intonations.
2. Gain modification: This modification is intended to prevent artifacts resulting from differences of mean signal amplitude between query and example ABs. Such differences occur when the same syllable is uttered in different intensities.
3. Query – example offset fine tuning: The query AB and its best matching example AB are unlikely to be aligned to one another. The correct alignment is found based on maximizing waveform correlation between these two matching ABs.
4. Cropping the desired part of the chosen example AB. As the hole to be filled is typically shorter than the chosen example, we do not necessarily need to inlay all of it. We therefore crop some part of the example AB and use it to fill the hole. The length of this part can be equal or greater to the hole's length, and is determined so that it minimizes artifacts.

Sections 4–7 describe our algorithm in detail.

4. Problem formulation

The problem we deal with involves an audio signal broadcast over an unreliable communication channel. Some data is lost on the way (see Fig. 1). This results in a pierced audio sequence, i.e. having temporal gaps. The original, unharmed signal $\mathbf{s}^{\text{org}}$ is a digital audio signal sampled at frequency f_s from an acoustic waveform. The received digital audio signal $\mathbf{s}^{\text{r}}$ is corrupted by missing data segments, but it also contains unpierced intact time spans.

As defined in Section 3, a temporal segment of samples in $\mathbf{s}^{\text{r}}$ is an AB. Each AB is L^{AB} samples long, corresponding to $N^{\text{packets}} \in \mathbb{Z}^+$ consecutive packets.[1] Then,

$$L^{\text{AB}} = N^{\text{packets}} L^{\text{packet}}. \tag{1}$$

The streaming signal $\mathbf{s}^{\text{r}}$ is divided on the fly into partly overlapping ABs, as depicted in Fig. 3. The overlap between two consecutive ABs is an integer number of packets,

$$N^{\text{overlap}} \in \left[0, \ldots, N^{\text{packets}} - 1\right]. \tag{2}$$

We choose $N^{\text{overlap}} = N^{\text{packets}} - 1$, to maximize the density of ABs. The overlap is therefore $L^{\text{overlap}} = N^{\text{overlap}} L^{\text{packet}}$ samples long.

[1] Restricting N^{packets} to integer values means that a packet is the smallest 'building block'. A smaller block contains too little information for signal analysis.

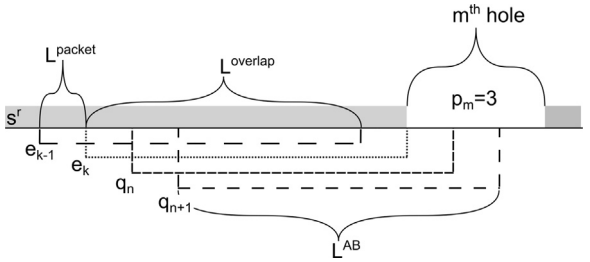

Fig. 3. Example AB extraction: examples $\mathbf{e}_k$ and $\mathbf{e}_{k-1}$ are intact ABs extracted from $\mathbf{s}^{\mathrm{r}}$, while query ABs $\mathbf{q}_n$ and $\mathbf{q}_{n+1}$ have missing portions.

4.1. Example AB

An undamaged AB is an example AB. The k^{th} example AB is denoted by $\mathbf{e}_k$. Let i_k index the first sample in $\mathbf{e}_k$. A sample of $\mathbf{s}^{\mathrm{r}}$ that corresponds to i_k is $\mathbf{s}^{\mathrm{r}}(i_k)$. Then,

$$\mathbf{e}_k = \left[\mathbf{s}^{\mathrm{r}}(i_k), \mathbf{s}^{\mathrm{r}}(i_k+1), \ldots, \mathbf{s}^{\mathrm{r}}(i_k+L^{\mathrm{AB}}-1)\right]. \tag{3}$$

Here $\mathbf{s}^{\mathrm{r}} = \mathbf{s}^{\mathrm{org}}$, since this AB is unpierced. Let $N_{\mathrm{E}}(\tau)$ be the number of unpierced ABs, which have appeared in the audio stream up to the current time τ. Then

$$\mathbf{E}_\tau = \{\mathbf{e}_k\}_{k=1}^{N_{\mathrm{E}}(\tau)} \tag{4}$$

is the set of *unpierced* example ABs, which have been captured up to this time.

4.2. Query AB

A hole is caused by at least one missing packet. Holes pierced in s_{r} are indexed by m, in order of appearance. There are usually less holes than missing packets, because some holes are created by a sequence of consecutive lost packets.

An AB that has some missing data is a query AB, denoted as $\mathbf{q}_n$ (see Fig. 3). Analogously to the definition in Eq. (3), let i_n index the first sample in $\mathbf{q}_n$. Then,

$$\mathbf{q}_n = \left[\mathbf{s}^{\mathrm{r}}(i_n), \mathbf{s}^{\mathrm{r}}(i_n+1), \ldots, \mathbf{s}^{\mathrm{r}}(i_n+L_{\mathrm{AB}}-1)\right]. \tag{5}$$

In a query AB, some samples are missing through their encapsulating packets. Let p_m be the number of consecutive missing packets that form the m^{th} hole. The number of consecutive missing samples, N_m^{samples}, in the m^{th} hole is then

$$N_m^{\mathrm{samples}} = p_m L^{\mathrm{packet}}. \tag{6}$$

These N_m^{samples} missing samples are equivalent to a gap in the audio signal, $N_m^{\mathrm{samples}}/f_s$ seconds long. For the purpose of brevity, the term 'packet' will be used to refer also to the segment of audio samples contained inside the packet.

For a query AB to be usable, some of its data needs to be intact (see Section 3). Therefore, we set the query length to be longer than the maximal probable hole length (see Section 2):

$$N^{\mathrm{packets}} > p_m. \tag{7}$$

The intact portions of $\mathbf{q}_n$ are denoted by $\mathbf{q}_n^{\mathrm{int}}$ (See Fig. 4). Our algorithm uses only $\mathbf{q}_n^{\mathrm{int}}$, since the data in other portions of $\mathbf{q}_n$ had been lost.

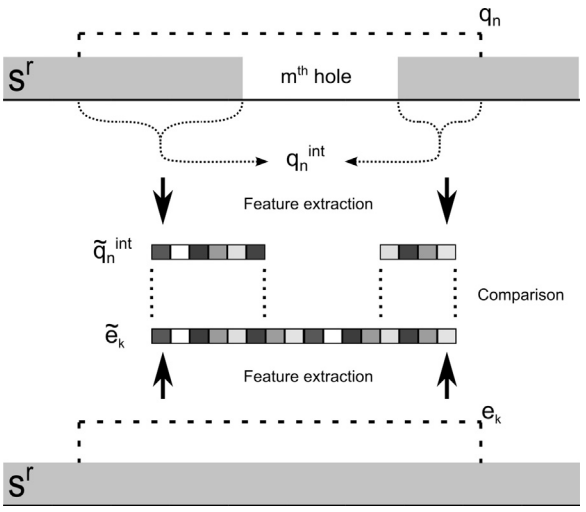

Fig. 4. Comparing a query AB to an example AB. The features corresponding to the intact query portions are in $\tilde{\mathbf{q}}_n^{\mathrm{int}}$. They are compared to the corresponding features in $\tilde{\mathbf{e}}_k$.

Each AB (either example or query) is pre-processed to yield audio feature vector:

$$\tilde{\mathbf{e}}_k = \mathcal{P}(\mathbf{e}_k), \quad \tilde{\mathbf{q}}_n = \mathcal{P}(\mathbf{q}_n^{\mathrm{int}}). \tag{8}$$

We used the *Mel frequency cepstral coefficients* (MFCCs) as features, since MFCCs are known to well express human perception of audio. The pre-process $\mathcal{P}$ that we use is described in further detail in Appendix A. The resulting example feature vectors comprise the set $\tilde{\mathbf{E}}_\tau$, corresponding to the set defined in Eq. (4).

5. Feature statistics as a prior

Before filling audio holes, we estimate the statistics of the unpierced signal, using training. The statistics then serves as prior knowledge when processing a pierced audio segment. A similar model was described earlier by Segev et al. in [13]. It is brought here with some necessary modifications.

When listening to a familiar language, a strong prior is that some temporal sequences of consecutive syllables are highly probable (frequently appearing in words), while others are much less so. The probability of a syllables temporal sequence is a prior knowledge, which can disambiguate speech under noise. Adopting this idea and abstracting it to become relevant to general audio streams, we propose to *avoid* high-level division of audio sequences into syllables. Instead, we use the low-level example ABs, and learn by training the transition probability between ABs, as we now describe.

The set of feature vectors $\tilde{\mathbf{E}}_\tau$ undergoes clustering into C clusters (using K-means), as illustrated in Fig. 5a. The proper number for C is debatable. For example, there are $\mathcal{O}(10^4)$ potential syllable types in speech, which roughly indicates C. To reduce dimensionality in our experiments, we take as a rule-of-thumb the number of vowel $\times$ consonant combinations (in any order), which leads to $C=300$. This way we obtain clusters of ABs that sound rather similar. ABs across clusters can efficiently be used in consecutive order to render speech. Let $\mathbf{e}_k$ belong to cluster $c_k = c(\tilde{\mathbf{e}}_k)$. We wish to learn the probability of temporal transition between ABs that are

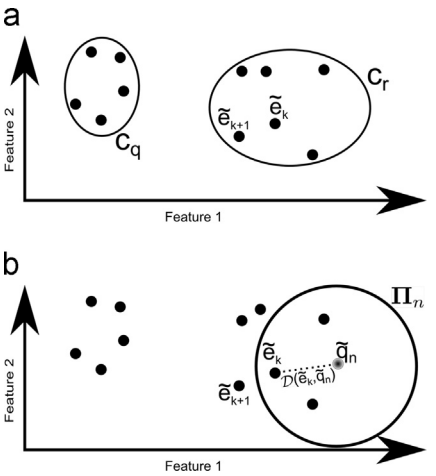

Fig. 5. Illustration of feature space, with two features. (a) Example of feature vector $\tilde{\mathbf{e}}_k$ is clustered, among other example feature vectors, into cluster c_r. (b) The subset Π_n includes the N^{cand} example ABs closest to $\tilde{\mathbf{q}}_n$. The distance to $\tilde{\mathbf{e}}_{k+1}$ is too large to be in Π_n.

conterminous (share a temporal boundary, with no overlap). The example AB that is conterminous to $\mathbf{e}_k$ is $\mathbf{e}_{k+N^{\text{stride}}}$, where $N^{\text{stride}} \in \mathbb{Z}^+$ is the difference of indices between two conterminous ABs. Using Eq. (2) and N^{packets} (defined in Section 4), we define[2]

$$N^{\text{stride}} \equiv \frac{N^{\text{packets}}}{N^{\text{packets}} - N^{\text{overlap}}}. \tag{9}$$

The set of all consecutive ABs corresponding to fixed clusters $q, r \in [1, \ldots C]$ is

$$\Phi_{q,r} = \left\{ k | c_k = q \quad \text{and} \quad c_{k+N^{\text{stride}}} = r \right\}. \tag{10}$$

The probability for a transition from cluster q to r is estimated from the histogram of these sets,

$$P(q,r) = |\Phi_{q,r}| / N_{\text{E}}(\tau). \tag{11}$$

In a $C \times C$ matrix $\mathbf{P}$, the (q, r) element is $P(q, r)$. This matrix is a statistical prior that expresses the joint probability for consecutive signal ABs. This prior models signals as derived from a degenerated hidden Markov model (HMM). In a general HMM, observations are emitted by hidden states. In our case, observations are ABs, and the hidden states are the clusters. However, each state (cluster) c_k has an exclusive subset of observations (ABs), assigned to it by the clustering procedure. In HMM terms, this means that the probability of ABs emitted by state c_k to be emitted by state $c_{r \neq k}$ is zero.

HMMs were used earlier in a related context [14–17]. In [14,15], HMM is used as the key component in a speech recognition task. However, our aim is different than speech recognition, as we aim at producing a plausible audio result. Hersheynand and Casey [16] used a more sophisticated HMM, incorporating some visual features, trying to enhance audio source separation performance for the purpose of speech recognition. Our problem is different than audio source separation, as we deal with sequences of missing audio data.

[2] We set the values of N^{packets} and N^{overlap} to satisfy $N^{\text{stride}} \in \mathbb{Z}^+$.

Rodbro et al. [17] incorporated HMM for the purpose of PLC as we do, but used it in a parametric approach as a generative model. Contrary to that, HMM in our method is not incorporated in a generative model, but rather used as a regularization mechanism. It supports an example-based approach, by regulating the choice of the best matching example-query pair, as explained in Section 6.2.2. Moreover, our method is indifferent to the phonetic and lingual semantics of audio signals, e.g. vowels, consonants, verbs and nouns. Instead, it uses previous self-content statistics as prior knowledge.

6. Example matching

We match each hole in $\mathbf{s}_{\text{r}}$ with its most appropriate example in $\mathbf{E}_\tau$, utilizing the unharmed data which surrounds the hole. This section elaborates on this process, which is done separately for each hole.

6.1. Query selection

Consider Fig. 6. The set of optional queries for the m^{th} hole is defined as

$$\mathbf{Q}_m = \{\mathbf{q}_n | m\text{th hole} \subset \mathbf{q}_n\}. \tag{12}$$

An AB query $\mathbf{q}_n$ can include more than a single hole (Fig. 6). In $\mathbf{Q}_m$, some queries contain more information than others, for the purpose of example matching. Within $\mathbf{Q}_m$, we prefer to use the more informative queries, which have a better chance to match a suitable example. Therefore, we employ pruning, yielding a subset of informative queries for the m^{th} hole, $\overline{\mathbf{Q}}_m \subseteq \mathbf{Q}_m$. The pruning process is described in Appendix B.

6.2. Defining a cost function

Now we seek to associate each query feature vector $\tilde{\mathbf{q}}_n | \mathbf{q}_n \in \overline{\mathbf{Q}}_m$ with an example feature vector $\tilde{\mathbf{e}}_k \in \tilde{\mathbf{E}}_\tau$. This association should satisfy two requirements:

1. The feature vectors $\tilde{\mathbf{e}}_k$ and $\tilde{\mathbf{q}}_n$ should be similar. This requirement is expressed by a *Data (fidelity) term* $\mathcal{D}$ in a cost function $\mathcal{C}$, defined next.
2. Consistency with prior knowledge. Based on $\mathbf{P}$, we derive the probability that $\mathbf{e}_k$ appears between the two ABs which adjoin $\mathbf{q}_n$ in s_{r}. This becomes a *Regularization term* $\mathcal{R}$ in $\mathcal{C}$, defined in the following.

From these two requirements, the cost to be minimized is

$$\mathcal{C}(\tilde{\mathbf{q}}_n, \tilde{\mathbf{e}}_k) = \mathcal{D}(\tilde{\mathbf{q}}_n, \tilde{\mathbf{e}}_k) + \lambda \mathcal{R}(\tilde{\mathbf{q}}_n, \tilde{\mathbf{e}}_k). \tag{13}$$

Here λ weights the regularization (prior) relative to the data term. We discuss λ towards the end of this section.

6.2.1. Data term $\mathcal{D}$

Similar feature vectors $\tilde{\mathbf{e}}_k$ and $\tilde{\mathbf{q}}_n$ indicate similarity[3] between $\mathbf{e}_k$ and $\mathbf{q}_n$. Hence, for each query feature vector $\tilde{\mathbf{q}}_n | \mathbf{q}_n \in \overline{\mathbf{Q}}_m$, a distance grade $\mathcal{D}(\tilde{\mathbf{q}}_n, \tilde{\mathbf{e}}_k)$ is calculated $\forall \tilde{\mathbf{e}}_k \in \tilde{\mathbf{E}}_\tau$ (See Fig. 5b). We use the Mahalanobis distance, as it accounts

[3] Recall from Section 4.2 that $\tilde{\mathbf{q}}_n$ is calculated using only $\mathbf{q}_n^{\text{int}}$. Hence vector similarity is measured using only $\mathbf{q}_n^{\text{int}}$ and its corresponding portions in $\mathbf{e}_k$.

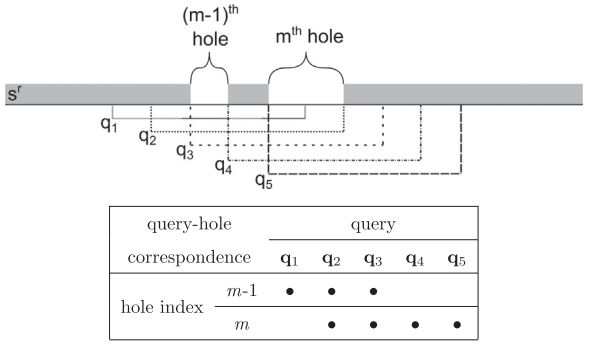

Fig. 6. Optional queries for the mth hole, when $N^{\text{packets}}=5$ and $p_m=2$. Note that queries $\mathbf{q}_2$ and $\mathbf{q}_3$ also contain the previous hole. Therefore, $\mathbf{q}_2$ and $\mathbf{q}_3$ appear in both sets $\mathbf{Q}_{m-1}$ and $\mathbf{Q}_m$. In the table, a bullet in a query-hole intersection indicates the query contains the hole in full.

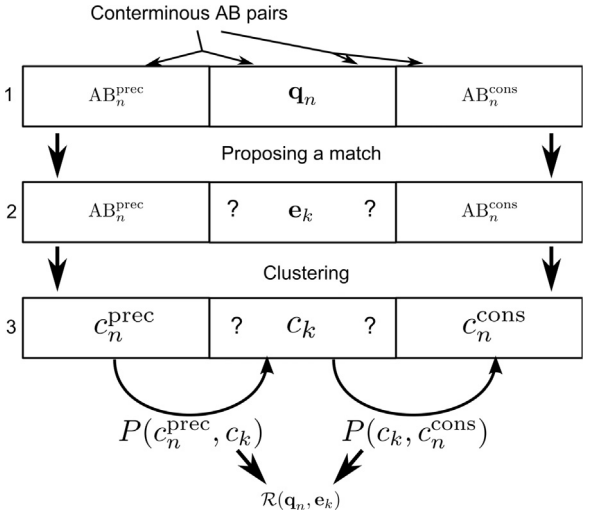

Fig. 7. Computing $\mathcal{R}$. Computed for query AB $\mathbf{q}_n$, that is conterminous to $\text{AB}_n^{\text{prec}}$ and $\text{AB}_n^{\text{cons}}$.

for correlation of features. The covariance matrix is estimated using $\forall\tilde{\mathbf{e}}_k\in\mathbf{E}_\tau$.

6.2.2. Regularization term $\mathcal{R}$

Query $\mathbf{q}_n$ is conterminous to preceding and consecutive ABs, denoted respectively by $\text{AB}_n^{\text{prec}}$ and $\text{AB}_n^{\text{cons}}$ (See Fig. 7). Suppose $\mathbf{q}_n$ is replaced by example $\mathbf{e}_k$. This yields a sequence of ABs:

$$\mathbf{AB}_{n,k}^{\text{seq}}=\left[\text{AB}_n^{\text{prec}},\mathbf{e}_k,\text{AB}_n^{\text{cons}}\right]. \tag{14}$$

This corresponds to a sequence of the clusters

$$\mathbf{c}_{n,k}^{\text{seq}}=\left[c_n^{\text{prec}},c_k,c_n^{\text{cons}}\right], \tag{15}$$

where c_n^{prec} and c_n^{cons} are the clusters of $\text{AB}_n^{\text{prec}}$ and $\text{AB}_n^{\text{cons}}$, respectively. This sequence has prior probability. A sequence of clusters is assumed to be a Markov process, hence

$$P(\mathbf{c}_{n,k}^{\text{seq}})=P(c_n^{\text{prec}},c_k)P(c_k,c_n^{\text{cons}}). \tag{16}$$

We use the Markovian nature of sequence $\mathbf{c}_{n,k}^{\text{seq}}$ to induce a cost:

$$\mathcal{R}(\mathbf{AB}_{n,k}^{\text{seq}})=-\log P(\mathbf{c}_{n,k}^{\text{seq}}). \tag{17}$$

From Eqs. (14), (16), (17), the regularization term is

$$\mathcal{R}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)=-\log P(c_n^{\text{prec}},c_k)-\log P(c_k,c_n^{\text{cons}}). \tag{18}$$

A low probability transition sequence between ABs induces a high cost, while a highly likely transition induces little cost.

Once the cost function terms are defined, finding the best match for the m^{th} hole yields a pair $(\mathbf{e}_m^{\text{best}},\mathbf{q}_m^{\text{best}})$. This pair comprises the example AB $\mathbf{e}_k\in\mathbf{E}_\tau$ which best matches query $\mathbf{q}_n\in\overline{\mathbf{Q}}_m$. The following list describes how this pair is found for the m^{th} hole:

1. Calculate $\mathcal{D}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)\ \forall\tilde{\mathbf{q}}_n|\mathbf{q}_n\in\overline{\mathbf{Q}}_m$ and $\tilde{\mathbf{e}}_k\in\tilde{\mathbf{E}}_\tau$, as in Table 1.
2. $\forall\tilde{\mathbf{q}}_n|\mathbf{q}_n\in\overline{\mathbf{Q}}_m$ keep the N^{cand} example feature vectors $\tilde{\mathbf{e}}_k\in\tilde{\mathbf{E}}_\tau$ with the smallest distance. This yields a subset Π_n of candidate pairs illustrated in Fig. 5b. It satisfies
$$\Pi_n=\{(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_j)|\mathcal{D}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_j)<\mathcal{D}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k),\forall k\notin\Pi_n\}. \tag{19}$$
The size of Π_n is N^{cand}.
3. Recall from Eq. (12) that the m^{th} hole has several optional queries. Each query $\mathbf{q}_n\in\overline{\mathbf{Q}}_m$ has a subset Π_n of candidate examples. Hence, we merge the subsets Π_n of $\forall\mathbf{q}_n\in\overline{\mathbf{Q}}_m$ to form a single complete subset of examples for the m^{th} hole:
$$\Pi_m=\bigcup_{n|\mathbf{q}_n\in\overline{\mathbf{Q}}_m}\Pi_n. \tag{20}$$
4. Calculate $\mathcal{R}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)$ for all pairs $(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)\in\Pi_m$, as illustrated in Fig. 7.
5. Using Eq. (13), obtain the best matching pair by
$$(\tilde{\mathbf{q}}_m^{\text{best}},\tilde{\mathbf{e}}_m^{\text{best}})=\underset{(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)\in\Pi_m}{\arg\min}\{\mathcal{C}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)\}. \tag{21}$$
6. The pair $(\mathbf{e}_m^{\text{best}},\mathbf{q}_m^{\text{best}})$ corresponding to the feature vectors pair $(\tilde{\mathbf{q}}_m^{\text{best}},\tilde{\mathbf{e}}_m^{\text{best}})$ is the solution.

The minimization of Eqs. (13) and (21) relies on λ. We used

$$\lambda=\overline{\lambda}\left(\underset{\tilde{\mathbf{q}}_n|\mathbf{q}_n\in\overline{\mathbf{Q}}_m,\tilde{\mathbf{e}}_k\in\tilde{\mathbf{E}}_\tau}{\text{median}}\{\mathcal{D}(\tilde{\mathbf{q}}_n,\tilde{\mathbf{e}}_k)\}\right). \tag{22}$$

Setting $\overline{\lambda}$ determines the balance between $\mathcal{D}$ and $\mathcal{R}$. We tested with several values of $\overline{\lambda}$ and empirically set it to 0.01.

7. Rendering an inpainted soundtrack

Following the minimization of $\mathcal{C}$, the matching pair $\mathbf{e}_m^{\text{best}}$ and $\mathbf{q}_m^{\text{best}}$ is found for the m^{th} hole. Consequently, we synthesize a restored audio signal $\hat{\mathbf{s}}$. The synthesis process is divided into several stages, aimed at reducing artifacts.

7.1. Pitch modification

Spoken syllables can be roughly divided into *unvoiced* and *voiced* [18]. Voiced syllables have a fundamental acoustic frequency (*pitch*). The pitch can vary between different occurrences of the syllable, due to intonation. Our example-matching algorithm is insensitive to intonation changes due to normalizations (described in Appendix A). Therefore, the

Table 1
Query-example distances. Values represent the distance between each $\tilde{\mathbf{e}}_k$ (rows) and each $\tilde{\mathbf{q}}_n$ (columns). For each query column, distance ranks appear in parenthesis. Per column, the N^{cand} elements having the smallest distance appear in bold.

Distance (rank#)		$\tilde{\mathbf{q}}_n \in \overline{\mathbf{Q}}_m$		
		$\mathbf{q}_{17}$	$\mathbf{q}_{19}$	$\mathbf{q}_{20}$
$\tilde{\mathbf{e}}_k \in \mathbf{E}_r$	$\tilde{\mathbf{e}}_1$	253 (#78)	**152 (#35)**	**124 (#31)**
	$\tilde{\mathbf{e}}_2$	486 (#320)	872 (#1053)	531 (#152)
	⋮	⋮	⋮	⋮
	$\tilde{\mathbf{e}}_{103}$	**116 (#35)**	515 (#334)	778 (#687)
	$\tilde{\mathbf{e}}_{104}$	576 (#325)	**60 (#7)**	**306 (#39)**
	⋮	⋮	⋮	⋮

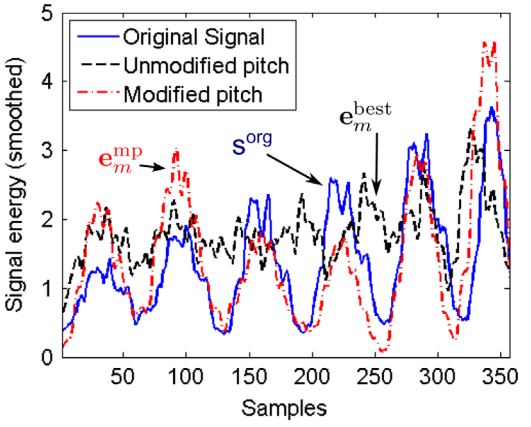

Fig. 8. Pitch modification of syllable 're'.

pitch of $\mathbf{e}_m^{\text{best}}$ can be inconsistent with $\mathbf{q}_m^{\text{best}}$. We assume that unlike the pitch itself, pitch variation within a syllable is similar across different utterances of the same syllable. Thus the pitch of $\mathbf{e}_m^{\text{best}}$ is modified by a *pitch modification* (PM) operator. To this end, we densely estimate the pitch of both $\mathbf{e}_m^{\text{best}}$ and $\mathbf{q}_m^{\text{best}}$ using autocorrelation. Then we use a phase vocoder implementation by [19], modifying pitch values throughout $\mathbf{e}_m^{\text{best}}$ to resemble the intact portions of $\mathbf{q}_m^{\text{best}}$, denoted $\mathbf{q}_m^{\text{best}-\text{int}}$. This yields a modified example $\mathbf{e}_m^{\text{mp}}$, whose pitch is more consistent with that of $\mathbf{q}_m^{\text{best}}$ (see Fig. 8):

$$\mathbf{e}_m^{\text{mp}} = \text{PM}(\mathbf{e}_m^{\text{best}} | \mathbf{q}_m^{\text{best}}). \tag{23}$$

We classify signal parts into voiced and unvoiced by thresholding the autocorrelation value, then modify only those classified as voiced.

7.2. Gain modification

Different occurrences of the same syllable also vary in their gain. As with intonation (Section 7.1), our example-matching algorithm is gain-invariant. Therefore, $\mathbf{e}_m^{\text{mp}}$ can have inconsistent gain with $\mathbf{q}_m^{\text{best}}$. For consistent gain, we amplify $\mathbf{e}_m^{\text{mp}}$ to match the energy of $\mathbf{q}_m^{\text{best}-\text{int}}$, yielding $\mathbf{e}_m^{\text{mpg}}$.

7.3. Timing fine tuning

Our algorithm uses a 'coarse to fine' approach. Recall from Section 6 that each hole is paired with a matching example. This match is temporally coarse. Now, the match is refined by temporally aligning $\mathbf{e}_m^{\text{mpg}}$ to the hole. To this end, we maximize a waveform correlation criterion by translating $\mathbf{e}_m^{\text{mpg}}$ relative to $\mathbf{q}_m^{\text{best}-\text{int}}$.

We primarily fill in the missing portion of $\mathbf{q}_m^{\text{best}}$. This missing portion corresponds to a certain portion in $\mathbf{e}_m^{\text{mpg}}$, denoted $\mathbf{e}_m^{\text{hole}} \subset \mathbf{e}_m^{\text{mpg}}$.

7.4. Optimal coupling

Synthesizing $\hat{\mathbf{s}}$ can apparently be done by replacing the pierced segments of $\mathbf{s}^{\text{r}}$ with $\mathbf{e}_m^{\text{hole}}$. However, this generally causes discontinuities in the waveform, resulting in annoying audible artifacts. In order to avoid these discontinuities, the transition between $\mathbf{s}^{\text{r}}$ and $\mathbf{e}_m^{\text{hole}}$ is done gradually, forming a transient phase. We use a linear weighting function to fade-out the signal $\mathbf{s}^{\text{r}}$, while fading-in $\mathbf{e}_m^{\text{hole}}$, and vice versa. In an attempt to further minimize discontinuities in $\hat{\mathbf{s}}$, we used optimal coupling [20] to determine the best transition timing (within a limited range) according to a spectral smoothness criterion [20].

8. Complexity analysis

As mentioned in Section 1, important criteria of a PLC method are its computational and memory costs, as they usually need to perform in real-time for maintaining a good VoIP quality. This paper aims at demonstrating an innovative PLC approach, and generally refrains from discussing implementation efficiency aspects. However, for providing a general idea of this method's computational costs, we bring here the *current* implementation's complexity analysis. A runtime comparison of this implementation with that of other baseline PLC methods is brought in Section 9.3.

As described in Section 3, our method comprises two mutual exclusive modes of operation, namely examples collection and data gaps handling. Several algorithmic stages are involved in each of the two. Their computational and memory costs analysis, based on [21,22], is presented in Table 2.

The current bottle necks of the proposed method's runtime are the data term $\mathcal{D}$ calculation and examples clustering, when incorporating regularization term $\mathcal{R}$. These bottle necks can be avoided by substituting these stages with more effective implementations, available today. The following is a brief list of several steps that can significantly improve the method's efficiency, thus rendering it capable of running in real-time:

1. Using an alternative online clustering method such as the ones in [23,24], that run in real-time, to substitute for the current clustering stage.
2. Using efficient methods such as in [25] to substitute for the nearest neighbors search (when calculating $\mathcal{D}$).
3. Using the real-time implementation of [21] to substitute for our feature extraction process.

Table 2
Computational steps and memory requirements (in words) of our different algorithm's steps. A bullet in columns EC and QH indicates the corresponding stage takes place in the Example Collection or Query Handling modes, respectively.

Algorithm's stage	Computational steps	Memory cost (Bytes)	EC	QH
Features extraction (MFCC)	$\mathcal{O}(N^{\text{MFCC}}+L^{\text{packet}})$	$\mathcal{O}(N^{\text{MFCC}}+L^{\text{packet}})$	•	•
Data term calculation, per hole	$\mathcal{O}(N^{\text{MFCC}}N^{\text{packets}}\cdot N_{\text{E}}(\tau))$	$\mathcal{O}(N_{\text{E}}(\tau))$		•
Examples clustering (K means)	$\mathcal{O}(\log N_{\text{E}}(\tau)\cdot N_{\text{E}}(\tau)^{(N^{\text{MFCC}}N^{\text{packets}}C+1)})$	$\mathcal{O}(N_{\text{E}}(\tau)+C)$	•	
Regularization term calculation, per hole	$\mathcal{O}(N^{\text{cand}})$	$\mathcal{O}(N^{\text{cand}})$		•
Rendering audio, per hole	$\mathcal{O}(N_m^{\text{samples}})$	$\mathcal{O}(N_m^{\text{samples}})$		•

4. Further saving computational load by relying on the network's guaranteed *quality of service* (QoS) parameter, reducing the density of extracted ABs or disabling our PLC mechanism completely when QoS is high.
5. Taking advantage of multi-thread capabilities, existing in many devices, to run computations on many example and query ABs simultaneously.

9. Experimental results

9.1. Tested audio

We simulated two VoIP scenarios. The spoken content of the first scenario is a text of a *story* [26] in English. We used a simple camcoder to record audio at 8 KHz, 313 s long. The recorded audio is $\mathbf{s}^{\text{org}}$. Then, a part at the beginning of $\mathbf{s}^{\text{org}}$, 30 s long, was pierced randomly according to the Gilbert model [12] to create $\mathbf{s}^{\text{r}}$. This model defines a probability to enter a packet dropping state and a probability to leave this state and return to normal packets delivery. In our experiment, we set the former to 0.06, and the latter to 0.11. The overall probability[4] of a packet to drop is 0.2. According to Ding and Goubran in [1], this simulates a typical packet dropping scenario. Parts of $\mathbf{s}^{\text{org}}$ that were not pierced at all simulate time periods without any packets dropped.

The second scenario constitutes a simulated telephone *conversation* between a salesman and a client, held using VoIP. We used this experiment to test our method's performance in such realistic scenario, that contains shorter sentences, embedded with silent parts. Here, example and query ABs were processed separately for the two conversing sides. We used a simple cellular phone to record audio at 8 KHz. We recorded 229 s, beginning with 109 unpierced seconds (used for ABs collection), followed by 2 min of audio sequence pierced according to the Gilbert model. Here we set the overall probability of a packet drop to 0.1.

[4] Calculation of the overall probability is slightly different than implied by [12] because we limit the number of consecutive dropped packets according to Eq. (7).

9.2. Tested PLC techniques

In our experiments, we compared different configurations of our proposed method with alternative PLC techniques. Two of the most commonly used methods were chosen for this comparison, namely ITU-G.711 and ITU-G.723.1. Additionally, we applied the method by Adler et al. [2]. The resulting audio sequences are available at [27]. The following methods and configurations were applied on the same pierced $\mathbf{s}^{\text{r}}$:

1. Silence: Trivially, audio gaps become periods of silence [5].
2. ITU-G.711: Applying the PLC mechanism according to this standard, using c language code supplied by the International Telecommunication Union [28].
3. ITU-G.723.1: Applying this modern PLC mechanism standard [29], designed specifically for VoIP applications. We used a Matlab implementation by Kabal [30].

We tested various configurations of our algorithm. They differ in the following parameters:

1. *Incorporation of* $\mathcal{R}$: Testing was performed with and without incorporation of prior knowledge as a regularization term in Eq. (13).
2. *Pitch modification*: In the story scenario, testing was performed with and without modifying the pitch in voiced cases, as discussed in Section 7.1.

In all configurations, $N^{\text{cand}}=40$ and $N^{\text{packets}}=7$. For the configurations incorporating $\mathcal{R}$, we used $\overline{\lambda}=0.01$. Seven (six) different audio sequences, each 30 (120) s long, were tested in the story (conversation) scenario:

1. The original unpierced $\mathbf{s}^{\text{org}}$, used as a reference.
2. The three baseline methods mentioned above [5,28,29].
3. Our method, in three (two, in the conversation scenario) different configurations.

Overall, 13 different audio sequences were tested in both scenarios.

9.3. Run-time

We compared the run time of our method's Matlab implementation with that of other baseline methods implemented in Matlab, when applied to the story scenario. We ran all methods on a computer equipped with an Intel Core i5-2520 M processor, 8 GB of RAM and a 64 bit "Windows 7 Ultimate" operating system. The comparison was conducted using the pierced part of the audio sequence described in Section 9.1, which is 30 s long.

The results show that the current implementation's run-time is roughly 10 times that of the ITU-G.723.1 method. Incorporating regularization term $\mathcal{R}$ further increases run-time by 40%. This run-time is too long for our method to be used as a PLC mechanism. However, as discussed in Section 8, several implementation modifications can be made, rendering our method capable of operating in real-time and reducing power consumption.

9.4. Objective results and subjective survey

Recall that our method's goal is to produce a perceptually plausible audio signal. Hence, its performance measure should accommodate this goal, by rating perceived audio quality. Degradation in perceived audio quality can be measured by the *mean opinion score* (MOS), defined in the ITU-T P.862 standard [31]. This score is widely accepted as a measure of speech quality assessment. Participants assess the quality of a given audio sequence by rating the level of its audio impairment from 'very annoying' (bad quality, grade 1) to 'imperceptible' (excellent quality, grade 5).

An objective estimation of the MOS was attempted with several approaches [32–35]. However, the MOS prediction capabilities in our tests were unsatisfying, as different PLC configurations' predicted scores did not differ significantly, regardless of their perceived quality. One such MOS prediction tool is the PESQ score, presented in Table 3 for both tested scenarios. It was computed using the SPDemo tool (available at [36]).

We therefore decided to conduct a subjective survey for the story scenario experiment. We used a listening test for assessing the performance of each of our method's configurations, compared to the baseline techniques. A set of 90 individual listeners from around the world listened and evaluated audio sequences according to the MOS scale. Participants were employed using the Amazon Mechanical Turk online service. Further details regarding the execution of this survey can be found in Appendix C. The method by Adler et al. [2] was excluded from our survey due to its rasping result, but its result is available for listening at [27].

Table 3
PESQ MOS predictions of the various tested PLC techniques and configurations, ordered by predicted MOS.

PLC method	Incorporating $\mathcal{R}$	Pitch modification	PESQ MOS Story	Conversation
Audio inpainting	✓	×	3.39	3.79
Audio inpainting	×	×	3.33	3.75
Audio inpainting	✓	✓	3.26	–
G.723.1	–	–	2.84	3.52
G.711	–	–	2.8	3.62
Silence	–	–	2.58	3.51
Inpainting with [2]	–	–	1.97	3.3

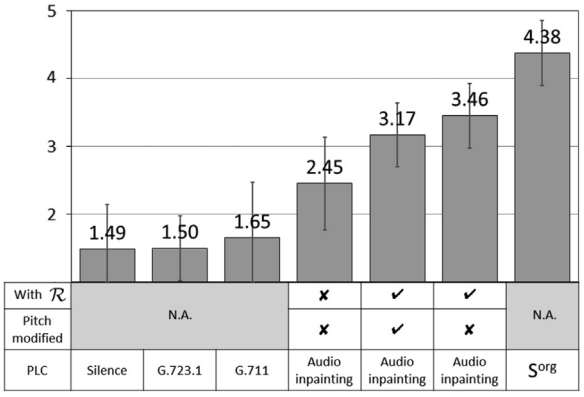

Fig. 9. MOS for different configurations of our audio inpainting method, compared with three common PLC methods, applied on the story scenario sequence. Margins of error are displayed at 95% confidence.

Each survey participant listened to six audio sequences, randomly chosen from the overall seven. Sequences were presented in random order, and each sequence was followed by an evaluation question. The survey results are presented in Fig. 9.

10. Discussion of results

In our story scenario test, the initial pierced audio had mean quality score between "Annoying" and "Very annoying". Our method improved the mean audio impairment score to between "Slightly annoying" and "Perceptible but not annoying". While tested on relatively long audio gaps (see Section 2), self-content-based inpainting by itself increased the MOS by approximately 1, compared to the baseline methods. Incorporating the regularization $\mathcal{R}$ (an HMM model) further increased the MOS to 3.5. In the conversation scenario test, objective MOS estimation indicates significant improvement compared to the baseline methods, when applying our method after less than 2 min of conversation.

The results in Fig. 9 and Table 3 give rise to several insights. The standard commonly used methods ITU-G.723.1 and ITU-G.711 demonstrated inferior performance. A possible cause is their mechanism of gradually muting their output after 20 ms of audio gap, producing utter silence after 60 ms [28,29]. This period of time is short relative to the 240 ms long gaps introduced in our test. A third commonly used standard, the ITU-G.722, possesses this same mechanism [37]. Due to this redundancy with ITU-G.723.1 and ITU-G.711, it was not included in our test.

Our method is indifferent to phonetic classes, as mentioned in Section 5. The method harnesses examples to fill gaps in a plausible manner, regardless of the phonetic and lingual characteristics. For instance, when listening to seconds 8–12 in all story experiments sequences but the last, the gap at the end of "green" occurs on a transition from a vowel to a consonant and the gap at the beginning of "they" occurs in the

beginning of a word. Both gaps are plausibly inpainted by our method. Lastly, pitch modification (Section 7.1) had no significant effect on the results, considering the margins of error.

11. Conclusions

The method presented here demonstrates potential of self-content-based inpainting, in the context of packet loss concealment. We fill-in data gaps using past recorded audio segments of the same speaker, thus reducing artifacts stemming from artificial speech synthesis.

The current implementation has several shortcomings that stimulate future improvements. One problem is reliance on past examples collected up to a gap. Therefore, the method lacks sufficient number of example ABs in early stages of a conversation. For that period, examples collected during previous conversations can be used. Another problem is run-time. It can be substantially reduced from our current implementation, as discussed in Section 8.

Due to the subjective nature of our method's objective, we evaluated its performance using an online survey, which requires a substantial number of participants for guaranteeing reliable evaluation. Since the number of survey participants increases in accordance to the number of different audio sequences compared, we only picked some representative subset of our method's configurations.

A future extensive subjective survey may examine other interesting aspects. These include the influence of different parameters (hole's length, size of $\mathbf{E}_\tau$, number of clusters C, etc.) and scenarios (phone conversation, different languages, environmental noise). Future research may also attempt using alternative audio features (MFCC derivatives, codec-dependent features), incorporating a higher lingual level HMM or even dynamically controlling L^{AB}, using the pitch estimation described in Section 7.1.

Acknowledgment

Yoav Schechner is a Landau Fellow, supported by the Taub Foundation. This research was supported by TASP – Technion Autonomous Systems Program and by the Israel Science Foundation (ISF) Grant 1031/08. It has also received funding from the European Research Council under European Union's Seventh Framework Program, ERC Grant agreement no. 320649. It was conducted in the Ollendorff Minerva Center. Minerva is funded through the BMBF.

We thank the reviewers for their constructive and insightful comments. We thank Antoine Deleforge for useful discussions and the lab group for heartily support. We thank Amir Adler for letting us use the code implementing his method [2]. Finally, we thank Oded Yeruhami for contributing his vocal speech to the recording and the online survey participants for their important contribution.

Appendix A. Audio features

MFCCs are commonly used as perceptual features of audio. Thus our features are based on MFCC. The following is a detailed description of the feature extraction process $\mathcal{P}$ mentioned in Eq. (8).

Speech is generally not stationary throughout the temporal extent of an AB. Therefore we calculate MFCCs for each packet separately. This yields N^{packets} MFCC vector sets $\{\mathbf{v}_t\}_{t=1}^{N^{\mathrm{packets}}}$ for each AB. Each row-vector $\mathbf{v}_t$ comprises N^{MFCC} coefficients, corresponding to N^{MFCC} frequency bands

$$\mathbf{v}_t = \{v_{t,b}\}_{b=1}^{N^{\mathrm{MFCC}}}. \tag{A.1}$$

The calculated MFCCs undergo a normalization process. This process is equivalent to performing cepstral mean normalization (*CMN*, see [38]), followed by taking the delta between each pair of consecutive MFCCs in the frequency domain. This process, applied for the purpose of improving the ability of a feature vector to represent ABs' similarity, consists of the following steps:

1. MFCCs are logarithmic values of the spectrogram. Therefore, biasing of MFCC provides gain adjustment. The *mean* value of each MFCC band, calculated over all the full audio sequence, is thus subtracted from this band's raw coefficients:
$$v_{t,b}^{\mathrm{IN}} = v_{t,b}^{\mathrm{raw}} - \underset{v_b \in \mathbf{s}^{\mathrm{r}}}{\mathrm{mean}}\{v_b\}. \tag{A.2}$$
This makes features gain insensitive.
2. For each frequency band coefficient, we subtract the preceding band's coefficient, yielding the final normalized version:
$$v_{t,b}^{\mathrm{Norm}} = \begin{cases} v_{t,b}^{\mathrm{IN}} & \text{if } b=1 \\ v_{t,b}^{\mathrm{IN}} - v_{t,b-1}^{\mathrm{IN}} & \text{if } b=2,\ldots,N^{\mathrm{MFCC}} \end{cases} \tag{A.3}$$
This makes features more sensitive to the packet's spectral shape, rather than to its coefficients' values.

Concatenating coefficients $v_{t,b}^{\mathrm{Norm}}$ for all frequency bands b yields the normalized coefficients vector for the packet in time t, $\mathbf{v}_t^{\mathrm{Norm}} = \left[v_{t,1}^{\mathrm{Norm}}, v_{t,2}^{\mathrm{Norm}}, \ldots, v_{t,N^{\mathrm{MFCC}}}^{\mathrm{Norm}}\right]$. Finally, concatenating these vectors for all packets of an AB yields the AB's normalized feature vector, mentioned in Eq. (8):

$$\tilde{\mathbf{AB}} = \mathcal{P}(\mathbf{AB}) = \left[\mathbf{v}_1^{\mathrm{Norm}}, \mathbf{v}_2^{\mathrm{Norm}}, \ldots, \mathbf{v}_{N^{\mathrm{packets}}}^{\mathrm{Norm}}\right]. \tag{A.4}$$

Appendix B. Query pruning

As mentioned in Section 6.1, the set $\mathbf{Q}_m$ of optional queries for the m^{th} hole undergoes pruning. Within $\mathbf{Q}_m$, some queries contain more information than others, for the purpose of example matching. This heterogeneity stems from two main reasons:

1. A query may have silence packets. In silence packets, the source of interest does not generate an audio signal. Therefore theses packets are dominated by noise. Classification of a packet as silence/non-silence is done by thresholding the signal's energy in the packet. This assumes that packets with high signal energy correspond to non-silence. We seek matches to the source of interest, hence we prune out silence packets.
2. Each query in $\mathbf{Q}_m$ has a certain number of missing packets. If two holes are close to each other (as in Fig. 6), some of the queries in $\mathbf{Q}_m$ contain (even partially)

Table 4
Paper notations table. Typical values appear in parenthesis.

Notation	Explanation
AB	Audio block
$\mathrm{AB}_n^{\mathrm{prec}}$	AB that precedes $\mathbf{q}_n$
$\mathrm{AB}_n^{\mathrm{cons}}$	AB that follows $\mathbf{q}_n$
$\mathbf{AB}_{n,k}^{\mathrm{seq}}$	ABs sequence induced by matching $\mathbf{q}_n$ and $\mathbf{e}_k$
C	Number of clusters (300)
$\mathcal{C}$	Cost function to minimize
c_n^{cons}	Cluster of the AB that follows $\mathbf{q}_n$
c_n^{prec}	Cluster of the AB that precedes $\mathbf{q}_n$
$\mathbf{c}_{n,k}^{\mathrm{seq}}$	Sequence of clusters induced by matching $\mathbf{q}_n$ with $\mathbf{e}_k$
c_r	A cluster indexed r
$\mathcal{D}$	Distance term in the cost function
$\mathbf{E}_\tau$	Set of example ABs collected up to time τ
$\mathbf{e}_k$	Example AB, indexed k
$\tilde{\mathbf{e}}_k$	Feature vector of the k^{th} example AB
$\mathbf{e}_m^{\mathrm{best}}$	Example AB that best matches the m^{th} hole
$\tilde{\mathbf{e}}_m^{\mathrm{best}}$	Feature vector that best matches the m^{th} hole
$\mathbf{e}_m^{\mathrm{mp}}$	Pitch-modified version of $\mathbf{e}_m^{\mathrm{best}}$
$\mathbf{e}_m^{\mathrm{mpg}}$	Pitch and gain-modified version of $\mathbf{e}_m^{\mathrm{best}}$
$\mathbf{e}_m^{\mathrm{hole}}$	Portion of $\mathbf{e}_m^{\mathrm{mpg}}$ that is inlayed into the audio signal
f_s	Audio signal sampling frequency, in Hz (8000)
L^{AB}	Number of samples contained in an audio block (2240)
L^{overlap}	Overlap between two consecutive ABs, in samples (1920)
L^{packet}	Number of audio samples contained in a single packet (320)
MOS	Mean opinion score (1–5)
N^{cand}	Num. of candidates kept per query (40)
$N_{\mathrm{E}}(\tau)$	Num. of example ABs collected up to time τ
N^{MFCC}	Number of MFCCs used (13)
N_m^{samples}	Number of missing samples forming the m^{th} hole (320–1920)
N^{overlap}	Overlap between consecutive audio blocks, in packets (6)
N^{packets}	Number of packets in an AB (7)
N^{stride}	Indices diff. between conterminous ABs (7)
$\mathbf{P}$	Probabilities matrix for transfer between HMM states
$\mathcal{P}$	Feature extraction process
p_m	Num. of packets causing the m^{th} hole (1–6)
$\mathbf{Q}_m$	Set of query ABs that contain the m^{th} hole
$\overline{\mathbf{Q}}_m$	Subset of valuable queries that contain the m^{th} hole
$\mathbf{q}_m^{\mathrm{best}}$	Query AB around the m^{th} hole optimally matching an example
$\mathbf{q}_m^{\mathrm{best-int}}$	Portions of $\mathbf{q}_m^{\mathrm{best}}$ that are *not* missing
$\tilde{\mathbf{q}}_m^{\mathrm{best}}$	Feature vector of the query AB around the m^{th} hole optimally matching an example
$\mathbf{q}_n$	Query AB, indexed n
$\mathbf{q}_n^{\mathrm{int}}$	Portions of $\mathbf{q}_n$ that are *not* missing
$\tilde{\mathbf{q}}_n$	Feature vector of the n^{th} query AB
$\mathcal{R}$	Regularization term in the cost function
$\hat{\mathbf{s}}$	Restored audio signal
$\mathbf{s}^{\mathrm{org}}$	The original unpierced audio signal
$\mathbf{s}^{\mathrm{r}}$	The received pierced audio signal
$\mathbf{v}_t$	MFCCs vector for packet at time t
$v_{t,b}^{\mathrm{Norm}}$	Normalized MFCC for freq. band b of packet at time t
λ	Weight of $\mathcal{R}$ in $\mathcal{C}$
$\overline{\lambda}$	Normalized weight of $\mathcal{R}$ in $\mathcal{C}$ (0.01)
$\mathbf{\Pi}_m$	Subset of all candidate examples to fill the m^{th} hole
$\mathbf{\Pi}_n$	Subset of closest N^{cand} examples to query $\mathbf{q}_n$

a neighboring hole ($m \pm 1$), while other queries are only pierced by the m^{th} hole itself. We generally prefer queries that have less missing packets, besides the m^{th} hole. For example, in Fig. 6, for the m^{th} hole, these are queries $\mathbf{q}_4$ and $\mathbf{q}_5$. They give more support for data comparison. Hence we prune out queries $\mathbf{q}_2$ and $\mathbf{q}_3$, which are pierced by a preceding hole.

Let $N_n^{\mathrm{significant}}$ be the number of packets in query $\mathbf{q}_n$ which are both classified as non-silent and correspond to non-missing packets. We define

$$\tilde{N}_m^{\mathrm{significant}} = \max_{\mathbf{q}_n \in \mathbf{Q}_m} \{N_n^{\mathrm{significant}}\}. \tag{B.1}$$

The pruned set of queries for the m^{th} hole is

$$\overline{\mathbf{Q}}_m = \{\mathbf{q}_n \in \mathbf{Q}_m | N_n^{\text{significant}} = \tilde{N}_m^{\text{significant}}\}. \quad \text{(B.2)}$$

In other words, $\mathbf{Q}_m$ is the subset of queries that share the maximal amount of significant data.

Appendix C. Amazon mechanical turk survey

In Section 9.4 we explain that a subjective survey was needed to evaluate our method's performance compared to other methods. We used the *Amazon Mechanical Turk* (AMT) service to refer participants to our online survey. Participants were employed by AMT and referred to our survey using a designated URL. Using AMT for this purpose brought up some issues which required attention. Following is a short review of some of them:

1. *Single participation*: In order to ensure survey credibility, each participant should only participate once. This limitation was enforced using a disposable survey URL. Each AMT worker received a different URL linking to our survey. Once it was used, it could no longer be used again for taking the survey.
2. *Motivating participants*: Possibly due to the single participation limitation, Participation was priced relatively high (around 80 cents, which induced an hourly wage of around 12 USD).
3. *Guaranteeing participants' attention*: In order to ensure the survey's credibility, we needed to make sure that the participant listened throughout the audio sequences. We accomplished this by informing participants of a content-related question to be presented at the end of the survey. Participants who failed to answer this trivial question correctly were excluded from the survey analysis. Another measure taken to prevent loss of participants' attention was limiting the number of sequences played to six, each 30 s long.

References

[1] L. Ding, R.A. Goubran, Assessment of effects of packet loss on speech quality in VoIP, in: Proceedings of the IEEE HAVE, 2003, pp. 49–54.
[2] A. Adler, V. Emiya, M.G. Jafari, M. Elad, R. Gribonval, M.D. Plumbley, Audio inpainting, IEEE Audio, Speech, Lang. Process. 20 (3) (2012) 922–932.
[3] B.W. Wah, X. Su, D. Lin, A survey of error-concealment schemes for real-time audio and video transmissions over the internet, in: Proceedings of the IEEE ISMSE, 2000, pp. 17–24.
[4] C. Perkins, O. Hodson, V. Hardman, A survey of packet loss recovery techniques for streaming audio, IEEE Netw. 12 (5) (1998) 40–48.
[5] J. Suzuki, M. Taka, Missing packet recovery techniques for low-bit-rate coded speech, IEEE J. Sel. Areas Commun. 7 (5) (1989) 707–717.
[6] E. Gunduzhan, K. Momtahan, Linear prediction based packet loss concealment algorithm for PCM coded speech, IEEE Audio, Speech, Lang. Process. 9 (8) (2001) 778–785.
[7] G. Zhang, W.B. Kleijn, Autoregressive model-based speech packet-loss concealment, in: Proceedings of the IEEE ICASSP, 2008, pp. 4797–4800.
[8] H. Ofir, D. Malah, I. Cohen, Audio packet loss concealment in a combined MDCT-MDST domain, IEEE Signal Process. Lett. 14 (12) (2007) 1032–1035.
[9] M. Lee, H. Kim, S. Choi, E. Lee, D. Kim, A forward-backward voice packet loss concealment algorithm for multimedia over IP network services, in: Proceedings of the Pacific-Rim Conference on Multimedia (PCM), vol. 3332, 2005, pp. 381–388.
[10] J. Lindblom, P. Hedelin, Packet loss concealment based on sinusoidal modeling, in: Proceedings of the IEEE Workshop on Speech Coding, 2002, pp. 65–67.
[11] W. Jiang, H. Schulzrinne, Modeling of packet loss and delay and their effect on real-time multimedia service quality, in: Proceedings of the NOSSDAV, 2000.
[12] G. Hasslinger, O. Hohlfeld, The Gilbert–Elliott model for packet loss in real time services on the internet, in: Proceedings of the IEEE Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008, pp. 1–15.
[13] D. Segev, Y.Y. Schechner, M. Elad, Example-based cross-modal denoising, in: Proceedings of the IEEE CVPR, 2012, pp. 486–493.
[14] S. Dupont, J. Luettin, Audio-visual speech modeling for continuous speech recognition, IEEE Trans. Multimed. 2 (3) (2000) 141–151.
[15] G. Potamianos, C. Neti, G. Gravier, A. Garg, A.W. Senior, Recent advances in the automatic recognition of audiovisual speech, Proc. IEEE 91 (9) (2003) 1306–1326.
[16] J. Hershey, M. Casey, Audio-visual sound separation via hidden Markov models, in: Proceedings of the NIPS, 2001, pp. 1173–1180.
[17] C.A. Rodbro, M.N. Murthi, S.V. Andersen, S.H. Jensen, Hidden Markov model-based packet loss concealment for voice over ip, IEEE Audio, Speech, Lang. Process. 14 (5) (2006) 1609–1623.
[18] J. Campbell, J.T. Tremain, Voiced/unvoiced classification of speech with applications to the U.S. government LPC-10E algorithm, in: Proceedings of the IEEE ICASSP, vol. 11, 1986, pp. 473–476.
[19] D.P.W. Ellis, A phase vocoder in Matlab, url: ⟨http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/⟩, 2002 .
[20] D.T. Chappell, J.H.L. Hansen, A comparison of spectral smoothing methods for segment concatenation based speech synthesis, Speech Commun. 36 (3) (2002) 343–373.
[21] J.C. Wang, J.F. Wang, Y.S. Weng, Chip design of MFCC extraction for speech recognition, Integr. VLSI J. 32 (1–2) (2002) 111–131.
[22] M. Inaba, N. Katoh, H. Imai, Applications of weighted voronoi diagrams and randomization to variance-based k-clustering, in: Proceedings of the Tenth Annual Symposium on Computational Geometry, 1994, pp. 332–339.
[23] Y. Chen, L. Tu, Density-based clustering for real-time stream data, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 133–142.
[24] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: Proceedings of the VLDB, vol. 29, 2003, pp. 81–92.
[25] S. Baluja, M. Covell, Audio fingerprinting: combining computer vision data stream processing, in: Proceedings of the IEEE ICASSP, vol. 2, 2007, pp. II-213–216.
[26] M.W. Brown, L. Weisgard, Red Light, Green Light, Scholastic Inc., 1992.
[27] Subjective Survey Results, URL: ⟨http://webee.technion.ac.il/~yoav/research/audio_inpaint.html.⟩.
[28] I. Telecommunication Union, ITU-TG.711 URL: ⟨http://www.itu.int/rec/T-REC-G.711/⟩.
[29] I. Telecommunication Union, ITU-T G.723.1, URL: http://www.itu.int/rec/T-REC-G.723.1/_page.print.
[30] P. Kabal, ITU-T G.723.1 Speech Coder: A Matlab Implementation, Technical Report, MMSP Lab Technical Report, Department of Electrical and Computer Engineering, McGill University, August 2011.
[31] International Telecommunication Union, ITU-T P.862, URL: ⟨http://www.itu.int/rec/T-REC-P.862/⟩.
[32] S. Voran, Objective estimation of perceived speech quality. Part I. Development of the measuring normalizing block technique, IEEE Speech Audio Process. 7 (4) (1999) 371–382.
[33] S. Voran, Objective estimation of perceived speech quality. Part II. Evaluation of the measuring normalizing block technique, IEEE Speech Audio Process. 7 (4) (1999) 383–390.
[34] T. Falk, W.Y. Chan, Hybrid signal-and-link-parametric speech quality measurement for voip communications, IEEE Audio, Speech, Lang. Process. 16 (8) (2008) 1579–1589.
[35] A. Raake, Short and long-term packet loss behavior: towards speech quality prediction for arbitrary loss distributions, IEEE Audio, Speech, Lang. Process. 14 (6) (2006) 1957–1968.
[36] Signal and Image Processing Lab, The Technion, Spdemo, URL: ⟨http://www-sipl.technion.ac.il/⟩.
[37] I. Telecommunication Union, ITU-TG.712. URL: ⟨http://www.itu.int/rec/T-REC-G.722-201209-I/en⟩.
[38] S. Furui, Cepstral analysis technique for automatic speaker verification, IEEE Trans. Acoust., Speech, Signal Process. 29 (2) (1981) 254–272.