Vocoder-side Voice Morphing for TTS
Product Requirements

Why Voice Morphing?

Recording new voices has huge cost.

Goal Hierarchy

1. No intelligibility penalty
2. High-quality (Naturalness)
3. Speaker similarity
Outline

● Background
● The Matching-Under-Transform problem
● The Matching-Minimization algorithm
● The Optimal-Dynamic-Frequency-Warping-and-Weighting algorithm
● Voice-Morphing algorithm design
● Results
Background

- Voice Conversion:
  - Traditional: Parallel corpora: [Stylianou], [Kain]
  - Modern: Non-parallel corpora: [Mouchtaris], [Erro], [Godoy], [Rosec], [Silen]
  - Adaptation HMM-based TTS: [Tokuda], [Zen], [Yamagishi], etc
    - speaker factorization, eigen-speakers

- GMM-based regression function: [Stylianou],[Kain]

\[ F(\vec{x}) = \sum_{k=1}^{K} p(k|\vec{x}) \left[ \vec{\mu}_{y,k} + \sum_{y,x}^{k} \sum_{x}^{k} \Sigma_{x}^{-1} (\vec{x} - \vec{\mu}_{x,k}) \right] \]
Background

- **Traditional approach** ([Moulines], early 90s)
  a. align spectra via DTW

- **Improved Traditional approach** ([Stylianou], mid 90s)
  a. align spectra via DTW
  b. convert spectra
  c. re-align converted spectra
  d. re-convert spectra
  e. iterate until convergence
Background

- **Matching Minimization:**
  - We do not need aligned utterances → replace DTW with Nearest Neighbor
  - NN matching error < DTW matching error

- **INCA algorithm:**
  a. match spectra via NN
  b. convert spectra
  c. iterate until convergence

**Bug:** DEGENERATE SOLUTION TERM
Problem: Matching-Under-Transform

Problem: Match the vectors of a dataset \( Y \sim P(Y) \) to the vectors of a dataset \( X \sim P(X) \) under a compensating transform \( Y = F(X) \).
Matching-Under-Transform

- Two sets of vectors: $X, Y$
- Parametric mapping: $X \rightarrow Y$
  - transform that compensates speaker differences
    
    \[ F(\mathbf{x}) = \mathbf{\beta} + A\mathbf{x} \]

- Non-parametric mapping: $Y \rightarrow X$
  - finds correspondences
    
    \[ p(\mathbf{x}_n | \mathbf{y}_q) \]

- Distortion criterion:
  
  \[ d(\mathbf{y}, \mathbf{x}) = (\mathbf{y} - F(\mathbf{x}))^T W (\mathbf{y} - F(\mathbf{x})) \]

- Global distortion:
  
  \[ D = \sum_q p(\mathbf{y}_q) \sum_n p(\mathbf{x}_n | \mathbf{y}_q) d(\mathbf{y}_q, \mathbf{x}_n). \]
Matching-Under-Transform

- Deterministic Annealing:
  \[ D' = D - \lambda H(X|Y) \]

- Optimizing non-parametric mapping (many-to-many):
  \[ p(\tilde{x}_n | \tilde{y}_q) = \frac{\exp\left\{ -\frac{1}{\lambda} d(\tilde{y}_q, \tilde{x}_n) \right\}}{\sum_i \exp\left\{ -\frac{1}{\lambda} d(\tilde{y}_q, \tilde{x}_i) \right\}}. \]

- Optimizing parametric mapping:
  - closed form solution

- When \( \lambda \to 0 \) the stochastic many-to-many mapping becomes a deterministic many-to-1 matching
Matching-Minimization

\[ p(\tilde{x}_n | \tilde{y}_q) = \frac{\exp\{-\frac{1}{\lambda}d(\tilde{y}_q, \tilde{x}_n)\}}{\sum_i \exp\{-\frac{1}{\lambda}d(\tilde{y}_q, \tilde{x}_i)\}}. \]

\( \lambda \to 0: \) Nearest Neighbors

**Matching-Minimization algorithm:** Iteratively optimizes parametric & non-parametric mappings until convergence:
1. optimize non-parametric mapping given transform
2. optimize transform given non-parametric mapping
Matching-Minimization

More expressive transforms:

GMM-based regression function:

\[ F(\tilde{x}) = \sum_{k=1}^{K} p(k|\tilde{x}) [\tilde{\mu}_k + \Sigma_k \tilde{x}] \]

For block-diagonal matrix (MCEP + Delta + Delta-Delta) with block \( \Sigma_k' \):

\[ \Sigma_k \tilde{x} = (\tilde{x}^T \otimes I_D) R \Sigma_k' \equiv X_n' \]

The solution is:

\[ \tilde{\gamma} = - \left( \sum_{q} p(\tilde{y}_q) \sum_{n} p(\tilde{x}_n|\tilde{y}_q) \Gamma_n T W_q \Gamma_n \right)^{-1} \left( \sum_{q} p(\tilde{y}_q) \sum_{n} p(\tilde{x}_n|\tilde{y}_q) \Gamma_n T W_q \tilde{y}_q \right). \]
Matching-Minimization

Matching-Minimization as an adaptation algorithm:

- Recovers both transform & matching (local minima)
- Non-parametric on the data: makes no assumptions regarding the underlying distributions in X, Y
- Parametric only on the transform
- Uses Mean-Squared-Error instead of likelihood

Other applications:
- Nearest-neighbor-like non-parametric adaptation for ASR: VTL, recording compensation, data denoising, near/far field, mixed narrowband/wideband training data
Optimal Dynamic Frequency Warping & Weighting

Morphed Source Spectral Envelope: \( \hat{T}_n(f) = S_n(w(f)) + B(f) \)

Error Criterion:

\[
\epsilon = \frac{1}{N} \sum_{n=1}^{N} \int_0^\pi (T_n(f) - S_n(w(f)) - B(f))^2 \, df
\]
Optimal Dynamic Frequency Warping & Weighting

Frequency warping function.

VTLN component.

Fine-tuning component.

Corrective filter: $E\{T(f) - S(w(f))\}$
Optimal Dynamic Frequency Warping & Weighting

**Optimize jointly** the frequency warping function and the frequency weighting in the *continuous* frequency domain using a variant of the Viterbi algorithm:

\[
\hat{w}, \hat{B}(f) : \arg\min_{w, B(f)} \epsilon
\]

- Frequency domain is discretized in many frequency bins.
- For each frequency bin, a frequency warping and weighting are estimated via a viterbi iteration.

- Viterbi is used to find the optimal path.
- Search neighborhood in black dots.
Voice Morphing Algorithm Design

- **Source speaker**: TTS corpus
- **Target speaker**: 10-150 utterances

1. **Alignment step**: match source/target speaker spectra
2. **Training step**: find optimal transform from source->target speaker

---

Google
Results: MOS of Morphed Speech

MOS-Naturalness

- Excellent: 5.00
- Good: 4.00
- Fair: 3.00
- Poor: 2.00

Maximum (saturation effect): 4.50

- ANDROID TTS (Q3 2014)
- ANDROID TTS (Q3 2015)

UPPER BOUND (Q3 2015):
- 3.50
- 3.60

UPPER BOUND (Q3 2014):
- 4.00
- 4.08

MORPHED ANDROID TTS (Q3 2015):
- 3.00
- 3.20

Confidential + Proprietary
### Results (EN-US, DEV Vocaine-LSTM)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22KHz PRODUCTION TTS: greco_barracuda_sample_rate_22050_en_us_sfg</td>
<td>3.798 ± 0.132</td>
</tr>
<tr>
<td>MORPHED VOCAINE-LSTM TTS: morph_lstm_en_sfg_dev_vctk_20150603_sfg_to_p362</td>
<td>3.794 ± 0.097</td>
</tr>
<tr>
<td>PRODUCTION TTS: greco_barracuda_sample_rate_16000_en_us_sfg</td>
<td>3.776 ± 0.117</td>
</tr>
<tr>
<td>MORPHED VOCAINE-LSTM TTS: morph_lstm_en_sfg_dev_vctk_20150603_sfg_to_p269</td>
<td>3.757 ± 0.099</td>
</tr>
<tr>
<td>VOCAINE-LSTM TTS: lstm_en_sfg_dev_20150603</td>
<td>3.737 ± 0.091</td>
</tr>
<tr>
<td>MORPHED VOCAINE-LSTM TTS: morph_lstm_en_sfg_dev_vctk_20150603_sfg_to_p330</td>
<td>3.723 ± 0.115</td>
</tr>
<tr>
<td>MORPHED VOCAINE-LSTM TTS: morph_lstm_en_sfg_dev_vctk_20150603_sfg_to_p244</td>
<td>3.693 ± 0.088</td>
</tr>
<tr>
<td>MORPHED VOCAINE-LSTM TTS: morph_lstm_en_sfg_dev_vctk_20150603_sfg_to_p351</td>
<td>3.677 ± 0.094</td>
</tr>
</tbody>
</table>
## Results (EN-US, DEV Vocaine-LSTM)

<table>
<thead>
<tr>
<th>Algorithm A</th>
<th>Algorithm B</th>
<th>Mean Score</th>
<th>Significant</th>
<th>preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morphed Vocaine-LSTM (p362)</td>
<td>Vocaine-LSTM</td>
<td>0.107 ± 0.101</td>
<td>A is significantly better than B</td>
<td>A (38.4%), B (30.5%)</td>
</tr>
<tr>
<td>Morphed Vocaine-LSTM (p362)</td>
<td>16kHz Barracuda</td>
<td>-0.191 ± 0.164</td>
<td>A is significantly worse than B</td>
<td>A (37.3%), B (45.9%)</td>
</tr>
<tr>
<td>Vocaine-LSTM</td>
<td>16 kHz Barracuda TTS</td>
<td>-0.351 ± 0.170</td>
<td>A is significantly worse than B</td>
<td>A (30.8%), B (50.4%)</td>
</tr>
<tr>
<td>Morphed Vocaine-LSTM (p269)</td>
<td>Vocaine-LSTM</td>
<td>0.096 ± 0.060</td>
<td>A is significantly better than B</td>
<td>A (39.1%), B (33.3%)</td>
</tr>
<tr>
<td>Morphed Vocaine-LSTM (p362)</td>
<td>22 kHz Barracuda</td>
<td>-0.346 ± 0.173</td>
<td>A is significantly worse than B</td>
<td>A (35.1%), B (51.9%)</td>
</tr>
<tr>
<td>Vocaine-LSTM</td>
<td>22kHz Barracuda TTS</td>
<td>-0.611 ± 0.170</td>
<td>A is significantly worse than B</td>
<td>A (26.8%), B (58.8%)</td>
</tr>
</tbody>
</table>
## Results - Real-Time Ratio

### Conditions
- en-US
- Nexus 7 2013 (deb)

<table>
<thead>
<tr>
<th>Device</th>
<th>Real-time ratio</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Morphing</td>
<td>With Morphing</td>
<td></td>
</tr>
<tr>
<td>deb, Normal speed (1x)</td>
<td>46%</td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>deb, Rapid speed (2x)</td>
<td>67%</td>
<td>76%</td>
<td></td>
</tr>
<tr>
<td>deb, Fastest speed (4x)</td>
<td>118%</td>
<td>136%</td>
<td></td>
</tr>
</tbody>
</table>
Examples - Changing voice character per word.

The Bear and the Dragon.

The Bear said "But I am just a bear. How can I fight a dragon?" with a deep voice.

The Wolf and the Dog (fable).

A gaunt Wolf was almost dead with hunger when he happened to meet a House-dog who was passing by.

- "Ah, Cousin", said the Dog.
- "I knew how it would be; your irregular life will soon be the ruin of you. Why do you not work steadily as I do, and get your food regularly given to you?"
- "I would have no objection", said the Wolf.
- "if I could only get a place."
Results

- Quality of morphed voices rivals quality of their source voices.

- Some morphed voices are **significantly better** than their source voices (Post-recording speaker correction?)

- Quality of best morphed voice gets pretty close to the quality of 16 kHz Barracuda TTS (current production TTS).
References

“The theory is simple but it is hard to implement.”

The only way to know whether you know the theory is to implement it.