Improving the Sampling of the Null Space of the Acoustic-to-Articulatory Mapping

Blaise Potard Yves Laprie *

LORIA-CNRS 615 rue du jardin botanique
54600 Villers-lès-Nancy FRANCE
E-mail: potard@loria.fr, laprie@loria.fr

Abstract

This paper presents a new method for sampling the null space of the acoustic-to-articulatory mapping, which is considerably faster and more accurate than the previous method presented by Ouni and Laprie [4]. This is achieved by using a simple stochastic exploration of the articulatory space instead of complex linear programming techniques. This new method allows for a much faster and more accurate inversion process.

1 Introduction

It is well known that one of the main difficulties of the acoustic-to-articulatory mapping is the nonuniqueness of the inverse mapping, where different vocal tract shapes can produce the same acoustics. The many-to-one nature of the inverse mapping was proven theoretically a long time ago, but evidence of it in real speech is still scarce [6].

The exploration of the null space consists in finding all articulatory vectors having the same acoustic image, in a small region of the articulatory space. This exploration is a key point of the codebook based inversion method developed by Ouni and Laprie [4].

In that method, inversion is performed using a piece-wise linear approximation of the articulatory-to-acoustic mapping computed using Maeda's articulatory model [2] and synthesizer [3].

The local linear approximation can be reversed by computing the pseudo-inverse, thus yielding a local linear acoustic-to-articulatory mapping. The linwhere F_0 is the acoustic image (a M-dimensional vector of formant frequencies, $M = 3$) of the center P_0, and $J_f(P_0)$ is a Jacobian matrix of the articulatory-to-acoustic mapping computed around P_0.

Therefore, each hypercuboid is characterized by its center P_0, its length vector $\vec{r}$, and its acoustic image by F_0 and $J_f(P_0)$.

2.2 Generation of inverse solutions

Given an acoustic vector F_x, the goal of acoustic-to-articulatory inversion is to find every P_x such that $f(P_x) = F_x$. This is done using a codebook lookup procedure.

For an articulatory vector P_x located in a hypercuboid $\mathrm{H_c}$, we can use the linear approximation relation: $f(P_x) \approx F_0 + J_f(P_0) \times (P_x - P_0)$.

Therefore, the acoustic-to-articulatory inversion amounts to solving the equation:

$$F_x = F_0 + J_F(P_0) \times (P_x - P_0), \qquad (2)$$

which can be rewritten as a purely linear equation:

$$b = A \times x, \qquad (3)$$

where $A = J_F(P_0)$, $x = P_x - P_0$, and $b = F_x - F_0$.

Using the Singular Value Decomposition (SVD) [1], it is possible to find all solutions to (3).

The SVD provides a particular solution x_0, and also an orthonormal base of the null space of matrix A – which is in our case a $(N - M)$-dimensional vectorial space.

We can thus write the general solution of (3):

$$x = x_0 + \sum_{j=1}^{N-M} \lambda_j v_j, \qquad (4)$$

where the v_j vectors form an orthonormal base of the null space, and the λ_j are arbitrary scalars. We therefore can find the general solution to (2):

$$P_x = P_{\mathrm{SVD}} + \sum_{j=1}^{N-M} \lambda_j v_j, \qquad (5)$$

where $P_{\mathrm{SVD}} = P_0 + x_0$.

The particular solution x_0 presents an interesting property: it has the smallest Euclidean norm among all possible solutions; consequently P_{SVD} is the closest to P_0 with regards to the Euclidean norm.

The approximation of eq. (2) is however only valid in the hypercuboid $\mathrm{H_c}$; therefore a solution P_x is only acceptable if $P_x \in \mathrm{H_c}$; it is thus necessary to compute the intersection of the N-dimensional hypercuboid $\mathrm{H_c}$ with the $(N - M)$-dimensional affine space of the solutions.

Unfortunately, in the general case, no known method exists for computing this intersection in a formal way. Ouni developed a heuristic method based on linear programming to sample this intersection.

Let α^i_{inf} and α^i_{sup} define the maximum and minimum values of the i^{th} articulatory parameter in $\mathrm{H_c}$ – i.e., $\mathrm{H_c}$ is the Cartesian product $\mathrm{H_c} = \prod_{i=1}^{N}[\alpha^i_{\mathrm{inf}}, \alpha^i_{\mathrm{sup}}]$. Then we have

$$\alpha^i_{\mathrm{inf}} \le P^i_{\mathrm{SVD}} + \sum_{j=1}^{N-M} \lambda_j v^i_j \le \alpha^i_{\mathrm{sup}}, i = 1..N, \qquad (6)$$

where v^i_j is the projection of the j^{th} basis vector of the null space onto the i^{th} articulatory parameter.

This system defines a 4-polytope, i.e., a bounded intersection of four half-spaces. To completely define this 4-polytope, we need to find all the extreme points of this domain, since the polytope solution is the convex hull of these points and determine the space contained in the polytope.

A two-step algorithm is used to approximate the intersection. In the first step the smallest 4-dimensional hypercuboid which contains the 4-polytope of solutions is determined by linear programming. The second step consists in sampling this four-dimensional hypercuboid and keeping samples that belong to $\mathrm{H_c}$. The detailed heuristic can be found in [4].

3 Sampling of the null space revisited

The sampling of the null space is definitely one of the weakest point in Ouni's method. It is very slow because of the linear programs, which take an especially long time in hypercuboids which contain no solutions. Furthermore, the sampling does not take into account the size of the solutions space, and simply tries to generate the same number of solutions in each hypercuboid, which leads to a very heterogeneous density of solutions from a hypercuboid to another (cf. Fig. 1 left). Finally, because of mutual compensations along some of the articulatory axes of the v_j vectors, a simple random sampling within a hypercuboid is not appropriate either, leading to a Gaussian repartition of the solutions along these articulatory axes around the P_{SVD} solution.

3.1 Choice of the initial solution

One of the first things we would like to improve is the time to decide whether a hypercuboid contains solutions. To this regard, one of the first observations one can make is that, although the property of the initial solution P_{SVD} – to be the closest to the center P_0 for the Euclidean norm – is indeed quite strong, it is not appropriate to decide whether a hypercuboid contains solutions. Let $d_{\mathrm{H_c}}$ be the articulatory norm defined as follows:

$$d_{\mathrm{H_c}}(x) = \max_{i=1..N} \frac{|x_i|}{r_i} \quad (7)$$

It is easy to verify that $P_x \in \mathrm{H_c} \Leftrightarrow d_{\mathrm{H_c}}(P_x - P_0) \leq 1$. For our elementary structure, it is thus clear that the most appropriate point for the initial solution would be the solution closest to P_0 according to the $d_{\mathrm{H_c}}$ norm.

Determining an exact solution minimizing this norm unfortunately requires linear programming. It is however straightforward to determine an approximate solution that "almost" minimizes the $d_{\mathrm{H_c}}$ norm using a stochastic exploration of the space of solutions, starting from P_{SVD}, iteratively sampling random solutions, and keeping it as a new starting position when closer to the center with regards to $d_{\mathrm{H_c}}$.

This yields a new initial point $P_{\mathrm{H_c}}$ that we will now use to generate solutions. A hypercuboid $\mathrm{H_c}$ will be considered to contain solutions if and only if $d_{\mathrm{H_c}}(P_{\mathrm{H_c}} - P_0) \leq 1$.

3.2 Generations of points

After having decided that a hypercuboid contains solutions, we now need to evaluate approximately the volume of the space of solutions $v_{\mathrm{H_c}}$, in order to get a homogeneous sampling in the whole articulatory space. We use a formula based on the position of $P_{\mathrm{H_c}}$ with regards to P_0 that appears to give fairly accurate results.

We then generate a number of solutions proportional to $v_{\mathrm{H_c}}$, doing random moves in the solution space from $P_{\mathrm{H_c}}$ and keeping only points belonging to $\mathrm{H_c}$.

3.3 Boundaries of the space to explore

Another problem we can identify on Fig. 1 (left) is an apparent Gaussian repartition of the solutions generated around the initial solution, and it would be preferable to have a more homogeneous distribution. Furthermore, we would like to be able to generate the solutions without having to compute the boundaries of the space to sample using linear programming as described in section 2.2.

To eliminate the apparent Gaussian repartition, the simplest solution is to explore a wider space than necessary. It appears that increasing the length of each boundary of the space to explore by about 30% is a fairly good compromise. Note that this slight modification increases the volume of the space to explore (and thus the average number of solutions to test) by about a factor of 3.

Finally, we wish to obtain accurate boundaries without using linear programming. We currently use a very fast heuristic: we simply use the reciprocal of the Euclidian norm of the vectors of the base of the null space, normalised by the radius of $\mathrm{H_c}$, i.e.:

$$|\lambda_j| \leq \frac{1}{\sqrt{\sum_{j=1}^{N} \left(\frac{v_j^i}{r_i}\right)^2}} [*1.3], j = 1..4 \quad (8)$$

4 Experiments

4.1 Homogeneity: qualitative and quantitative results

To illustrate qualitatively the improvements brought by this method, we performed the static inversion of isolated vowels, and display the solutions generated with both methods. In this experiment we generated a vast number of solutions: about 200.000 solutions for each vowel; for practical purposes, e.g. inversion of speech sequences, we typically generate only between 1000 and 5000 solutions for each speech sample.

Fig. 1 illustrates the distribution of inverse solutions (for vowel /i/) with both methods, projected along two articulatory dimensions. It can be seen that the sampling effect (the vertical lines, and in a lesser extent the horizontal lines) vanishes with our new method.

To measure quantitatively the effects of having a more homogeneous generation of solutions, we conducted inversion experiments on sequences of speech for which we had the corresponding articulatory trajectories, and we compared the solutions found to the original one.

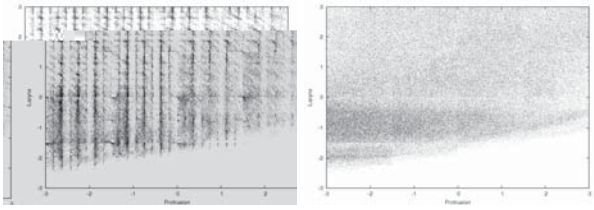

Figure 1: *Comparisons of the solutions generated by Ouni's (left) and our (right) method for the inversion of vowel /i/. Solutions are presented along two articulatory dimensions: lip protrusion and larynx height.*

nbsol	New		Ouni	
	$d_{\min}$	d_{avg}	$d_{\min}$	d_{avg}
1000	0.349	1.465	0.354	1.449
3000	0.300	1.468	0.323	1.482
10000	0.265	1.471	0.279	1.484
30000	0.229	1.473	0.239	1.485
100000	0.205	1.475	0.211	1.485

In this table, $d_{\min}$ is the minimal Euclidean articulatory distance to the original trajectory among the articulatory vectors generated; d_{avg} is the average Euclidean articulatory distance.

From this table, we can observe a significant decrease of this distance in the case of the new method compared to Ouni's, for the same number of solutions sampled; on average we needed to sample 70% more points with Ouni's method than with ours to get the same accuracy. This demonstrates a signifant increase in the quality of the sampling.

4.2 Computation time

The previous method was using linear programming to determine the boundaries of the space to explore, which took a very long computation time, especially when there was no solution inside the hypercuboid.

This problem does not happen anymore for two reasons: the new initial point used allows us to immediately decide if the intersection is empty or not; therefore, we would not need to use linear programming for the cases where there is no solution. Second, we do not use linear programming anymore for the exploration.

It is still interesting to measure which element brings the most improvement in computation time; we thus measured time to perform inversion in three different conditions: with Ouni's method; with the new heuristic to determine whether an hypercuboid contains solutions but with Ouni's exploration; with the complete new method.

Inversion experiments in the three different conditions show that a considerable amount of time was lost doing linear programming in hypercubes containing no solutions, since inversion in the second condition is already faster than Ouni's method by a factor 7. Furthermore, we can see that dropping linear programming altogd [(In)254(llo)23(w)-8(s)-3r additional factor 6 to be gained in computing time, and overall a factor 40 over Ouni's method.

5 Conclusion

We presented a considerably faster method to explore the null space of the acoustic-to-articulatory mapping, which also noticeably increases the quality of the sampling of solutions. We should point out hwsever that for practical purposes, e.g. the inversion of speech sequences, such raffinments are mainly not needed: indeed, the number of solutions generated in each hypercuboid has to be very strictly controlled, or the subsequent steps of the inversion process would have an overwhelming complexity. In these cases, we typically generate less than 10 solutions in each hypercuboid; since there is almost no sampling of the null space, the accuracy has no real influence, and therefore the most interesting characteristic is that we can decide very rapidly whether a hypercuboid contains solutions, which considerably reduces the computation time.

References

[1] G. Golub and C. V. Loan. *Matrix Computations*. JohnHopkins Universitys, Baltimore, 1989.

[2] S. Maeda. Un modèle articulatoire de la langue avec des composantes linéaires. In *Actes 10èmes Journées d'Etude sur la Parole*, pages 152–162, Grenoble, Mai 1979.

[3] S. Maeda. A digital simulation of the vocal tract system. *Speech Communication*, 1:199–229, 1982.

[4] S. Ouni andwY. Laprie. Modeling the articulatory space using a hypercube codebook for acoustic-to-articulatory inversion. *Journal of the Acoustical Societyof America* , 118(1):444–460, 2005.

[5] B. Potard and Laprie. Compact representationof the articulatory-to-acoustic mapping. In *Interspeech, Anvers*, pages 2481–2484, Aug. 2007.

[6] C. J 4 2in and M. A. Carreira-Perpiñán. An empirical investigation of the nonuniqueness in the acoustic-to-articulatory mapping. In *Interspeech, Anvers*, pages 74–77, Aug. 2007.