말소리와 음성과학 제1권 제3호 (2009)
pp. 73~85

Synthesis and Evaluation of Prosodically Exaggerated Utterances

Yoon, Kyuchul[1)]

ABSTRACT

This paper introduces the technique of synthesizing and evaluating human utterances with exaggerated or atypical prosody. Prosody exaggeration can be implemented by manipulating either the fundamental frequency (F0) contour, the segmental durations, or the intensity contour of an utterance. Of these three prosodic elements, two or more can be exaggerated at the same time. The algorithms of synthesis and evaluation were suggested. Learner utterances exaggerated in each of the three prosodic features were evaluated with respect to their original native versions in terms of the differences in their F0 contours, the segmental durations, and the intensity contours. The measure of differences was the Euclidean distance metric between the matching points in their F0 and intensity contours. The measure was calculated after the exaggerated learner utterances were aligned by the segments and rendered identical to their native version in terms of their segmental durations. For the evaluation of the segmental durations, no prior modifications were made in durations and the same measure was used. The results from the pilot experiment suggest the viability of this measure in the evaluation of learner utterances with atypical prosody with respect to their native versions.

Keywords: speech synthesis, prosody evaluation, prosody, fundamental frequency, segmental durations, intensity, Euclidean distance, Praat, English

1. Introduction

The importance of prosody in the production and perception of speech is well-known. Acquiring the correct prosody of a target language is even more important in learning second languages in order to reach the level of fluency of the native speakers. Compared with the learning of the segmental aspects of a language, learning the prosodic aspects can be more difficult. It may not be easy for teachers to find the right tools to teach the target prosody to learners of the second language. It is also possible that the theories of the prosody for the target language are not very “learner-friendly”, leaving language teachers to resort to the simpler but inadequate intonation patterns found in the traditional grammar books.

The prosody of a language can be regarded as a combination of at least three physical components. The fundamental frequency (i.e. F0) contour, also known as the intonation contour, is maybe the most emphasized of the three components in language education. Many students even identify the F0 contour with the prosody, which of course is not true. The second component is the segmental durations, which are also known to contribute to the prosody of an utterance. The phrase-final lengthening is one of such cases where the segmental durations play an important role. The third component is the intensity contour, generally known as the contour of loudness of an utterance.

To a greater of lesser degree, all these three components are regarded as contributing to the overall prosodic pattern of an utterance in a language. The roles and the interaction of these prosodic components are a good source of research for many linguists. The findings from such research can be reflected in language education. For this to happen, one needs to be able to manipulate each of the prosodic components to suit one's needs. It would also be necessary to manipulate two or all three

1) This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2007-327-A00310).

Received: July 9, 2009
Revision received: August 7, 2009
Accepted: August 23, 2009

components at the same time. If language teachers were able to manipulate the prosodic components at their will, students would be more aware of the importance of the prosody in learning a second language.

The ability to resynthesize given utterances with different prosody can be a useful tool for language teachers. Repeating the same utterance can be an inefficient way to teach language prosody. If teachers were able to emphasize a certain aspect of the prosody, for example the F0 contour of some parts of an utterance, learners would be more aware of that aspect of the prosody. In terms of the speech synthesis, the "emphasizing" can be realized in different ways. One way would be to "exaggerate" the particular prosodic aspect of an utterance. If it were the F0 contour, some parts of the utterance could be made higher than the original F0 contour.

One purpose of this paper is to introduce the technique of manipulating one or more of the prosodic components of an utterance. The technique was implemented as a script in Praat [1]. The technique of manipulating the F0 contour and the segmental durations is already available thanks to the work in [2]. However, it is not easy for language teachers to apply the technique to recorded utterances automatically. Moreover, the technique of manipulating the intensity contour is well beyond the reach of ordinary language teachers. Thus, the script tool to be introduced in this paper can be a valuable tool for language teachers in synthesizing utterances with exaggerated prosody and thus in teaching the prosody of the target language efficiently.

The prosody manipulation is closely associated with the evaluation of the prosodic aspects of an utterance. The technique can be used to design a series of experimental stimuli in the examination of measures proposed in the evaluation. For example, the technique of synthesizing utterances with exaggerated prosody introduced in this paper can be used to create a set of incrementally exaggerated stimuli. By comparing the original utterance produced by the native speaker and the exaggerated stimulus synthesized by the technique, one can test the viability of using some measure in the prosodic evaluation of the utterances involved.

The other purpose of this paper is to use the technique of prosody manipulation in order to test the viability of using the Euclidean distance metric in the prosodic evaluation of the utterances involved. The evaluation will be made in terms of the three prosodic features, producing three separate scores for a pair of utterances. The viability of each of the three scores will be tested with a pilot experiment.

2. Methods

2.1 Exaggerating prosody

Depending on how you define the term "exaggeration", prosody can be exaggerated in different ways. Given an utterance from a native speaker of the target language (See <Figure 1>), the F0 contour of the utterance can be exaggerated either by making all the F0 peaks higher or by making all the F0 valleys lower or by moving the peaks and the valleys at the same time. The peak can also be defined by specifying the frequency difference between a neighboring peak and a valley. The evaluation of the perceptual effects of these procedures could be performed with a perception experiment. Since it is not the purpose of this paper to do experiments to evaluate the perceptual effects, it suffices to say that F0 contours can be exaggerated by manipulating the F0 peaks and/or valleys of a recorded utterance.

Contrary to what can be done to exaggerate the F0 contour, it appears that there is only one direction in terms of exaggerating the segmental durations, that is, making segments longer. The well-known phrase-final lengthening can be regarded as one form of exaggerating segmental durations. In this case, the exaggeration works by making the final syllable of the phrase longer, thereby signaling the end of a phrasal organization of an utterance. There does not seem to be any linguistic device in a normal utterance that makes some part of an utterance shorter than usual to achieve any perceptual goal.

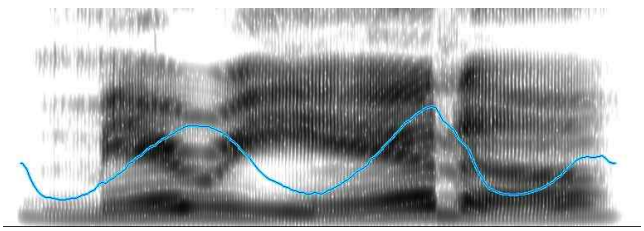

Figure 1. The fundamental frequency (F0) contour of an utterance *Marianna!*. The vertical axis for the F0 contour is in Hz. The F0 contour can be said to have two peaks and three valleys.

The next thing to consider in terms of exaggerating the segmental durations is where to exaggerate. The rules of thumb here have much to do with the F0 contour. It is beyond doubt that stressed syllables tend to become longer in durations. Moreover, English stress is usually associated with higher F0 peaks at the word level. Thus, stressed syllables are produced longer in durations and with higher F0 peaks. Therefore, duration exaggeration can be defined as making longer the region that has already been exaggerated by the F0 manipulation, i.e. the F0

peaks. The manipulation of the F0 contour and the segmental durations can be achieved by using the PSOLA algorithm proposed in [2]. The algorithm implemented in Praat enables one to manipulate the two prosodic components. However, the task is a laborious one and not suitable for regular language teachers to handle.

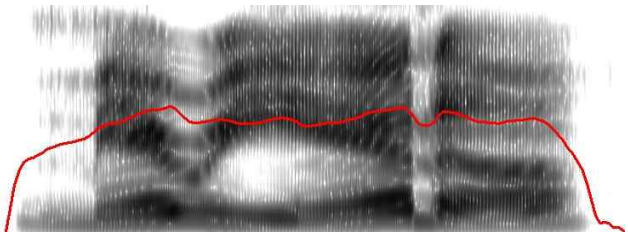

Figure 2. The intensity contour of an utterance *Marianna!*. The vertical axis for the intensity contour is in dB.

The intensity contour, as shown in <Figure 2>, can be exaggerated either by making the intensity peaks higher or by making the intensity valleys lower or by moving the peaks and valleys at the same time. The manipulation of the intensity contour can be achieved by manipulating the *IntensityTier* object in Praat. A user can add or remove intensity points to the object over time.

As suggested above, prosody exaggeration can be implemented in different ways depending on how you see exaggeration. Whichever method one may choose, one can only know how listeners perceive the exaggeration by performing a series of perception experiments. The technique of exaggerating prosody automatically with the help of a computer script can be a valuable tool in such experiments. Even if the user is not interested in the perceptual aspects of the prosody, this technique can be used as a practical tool in teaching prosody to learners of other languages.

2.2 The algorithm

<Figure 3> shows the parameters that can be set by the user. The dialog box pops up when the user runs the Praat script (See Appendix). Of the three prosodic components, i.e. the F0 contour, the segmental durations and the intensity contour, users can choose which one(s) to exaggerate by clicking the radio button(s) on the dialog box. If the user is only interested in seeing the effect of exaggerating the F0 contour, she can do so by clicking *Yes* of the *ExaggerateFrequency* item and clicking *No* to the other two components. Any combination of the three prosodic components should work fine.

In the first section on the F0 contour exaggeration, there is a

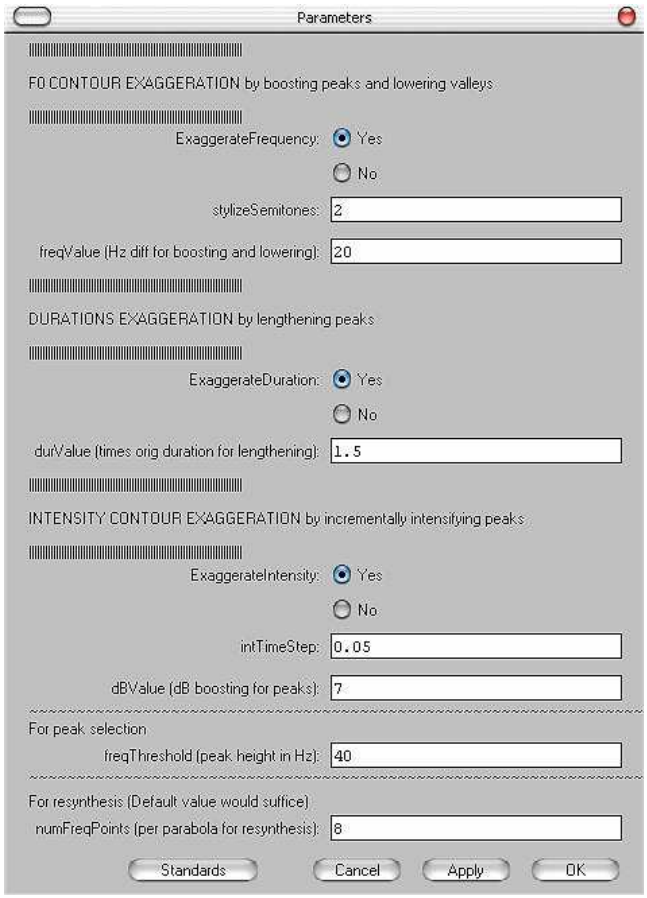

Figure 3. Parameter settings for the Praat script. The pop-up window is composed of three sections for the F0 contour, durations, and intensity contour exaggeration.

short explanation of what it means to exaggerate the F0 contour. F0 exaggeration was defined in the current work as making the F0 peaks higher and the F0 valleys lower. Users can manipulate the F0 peaks and valleys by specifying the *freqValue* variable in the section. By default, the script will move the peaks and valleys by 20Hz, meaning that the F0 peaks will be 20Hz higher and the F0 valleys 20Hz lower than their original height. The identification of the F0 peaks and valleys is based on the F0 contour stylization algorithm implemented in Praat. The frequency resolution of the algorithm is set as 2.0 semitones.

Here is how the F0 exaggerating algorithm works. Given the stylized version of the F0 contour of the original utterance (See <Figure 4>), the script starts comparing two neighboring pitch points. If the F0 difference between the two pitch points is greater than the threshold, specified in the *freqThreshold* variable at the bottom of the dialog box in <Figure 3>, the pitch point with a higher F0 value gets boosted by the amount specified in the *freqValue* variable, while the pitch point with a lower F0 value gets lowered by the same amount. If the F0 values are the same for the two pitch points, nothing happens. The process is

repeated with the next two pitch points, the first of which is the second pitch point from the previous step. Thus, in every step, there is one overlapping pitch point. The reason for the existence of the threshold frequency is that we do not want any minor "bumps" in the stylized F0 contour to be exaggerated. With this threshold, we can dictate how high an F0 bump has to be to be called a legitimate peak.

The stylized F0 bumps in <Figure 4> can all be called legitimate F0 peaks because the differences between the peaks and the valleys are more than the F0 values specified in the *freqThreshold* variable. Thus the two peaks will be made higher and the valleys lower. After the exaggeration of the peaks and the valleys, the manipulated, but still stylized, F0 contour is made natural by smoothing the contour using the parabola function. The number of points per parabola is specified in the *numFreqPoints* variable (8 by default).

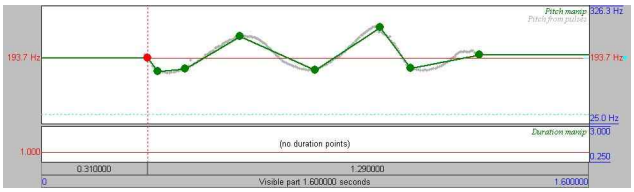

Figure 4. The stylized F0 contour of the utterance *Marianna!*.

Duration exaggeration is defined as lengthening the stylized pitch peaks by the amount specified in the *durValue* variable (1.5 by default) in the second section of the pop-up dialog box. The neighboring valleys remain the original duration. In other words, as shown in <Figure 5>, the segmental durations on the uphill of a pitch peak will be increased linearly by 1.5 times the original durations, while those on the downhill of the peak will be decreased linearly by the same amount. The durations near the pitch valleys are not shortened but remain the same as the original values.

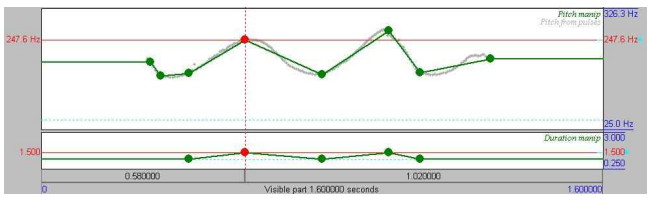

Figure 5. The stylized F0 contour of the utterance *Marianna!* (Upper window). The lengthening of the pitch peaks (by 1.5 times) for the duration exaggeration (Lower window).

In order to avoid the situation where the minor pitch bumps are involved in the exaggeration of the segmental durations, the pitch bumps need to be screened with the *freqThreshold* value to be identified as legitimate peaks. As mentioned above in the section on the F0 contour exaggeration, a pitch bump can be called a legitimate peak only if it is high enough. The frequency threshold determines this height. If the height of either the uphill or the downhill of the pitch bump is greater than the frequency threshold, the bump can be called a peak. Only these peaks are lengthened or exaggerated.

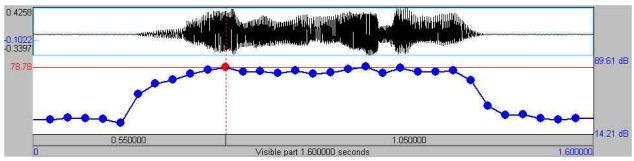

Figure 6. The intensity contour of the utterance *Marianna!* as represented in the *IntensityTier* object in Praat. The round dots represent the intensity points of the object. The intensity values of the points are in dB.

Exaggerating the intensity contour is defined as incrementally increasing the intensity values toward the legitimate pitch peaks. For the utterance in <Figure 6>, the legitimate peaks to be exaggerated correspond to the stressed syllables of the word *Marianna*. However, note that the locations of the peaks and their neighboring valleys do not come from the intensity contour given in <Figure 6>, but from the F0 contour given in <Figure 4>. That is because the intensity exaggeration was defined in terms of the pitch peaks, not in terms of the intensity peaks. The third pitch point in <Figure 4> becomes the immediately neighboring valley pitch point of the first legitimate pitch peak, which falls on the fourth pitch point. The user can also specify the interval between intensity points by setting the *intTimeStep* variable (0.05sec by default).

The exaggeration of the intensity contour proceeds in between the two locations, i.e. the uphill between the valley and its immediately following peak. The *dBValue* variable (7 by defaults) specified by the user is divided by the number of intensity points between the two locations. The increment for each intensity point is accumulated until the intensity point at the peak is reached. If, for example, the *dBValue* is 14dB and there are seven intensity points (See <Figure 6>) between the valley and the peak, the increment will be 2 dB per intensity point. Then, the accumulated values for each of the seven intensity points will be 2, 4, 6, 8, 10, 12, 14 dB's respectively. Thus, the closer the intensity points toward the peak, the greater become the accumulated intensity values. That is, we do not add a single uniform intensity value to all the intensity points to be exaggerated, but varying intensity

values to them. Therefore, the *dBValue* specified by the user is the maximum intensity value that the intensity point at the peak would get in the process of intensity exaggeration. The reverse process is done for the downhill between the peak and its immediately following valley.

The algorithm of intensity exaggeration is not based on any perception experiments, but on trial and errors. Adding a uniform value to the intensity points would result in an unnatural intensity contour. It is reasonable to think that the intensity contour should increase or decrease without any audible "jumps". Gradual increase of the intensity values is one way to achieve this goal. It may be a good idea to consider non-linear mathematical functions for the manipulation of intensity points in the future.

2.3 Evaluating exaggerated prosody

The evaluation here refers to the comparison of the exaggerated or atypical utterance with its non-exaggerated or typical original utterance. As in the exaggeration, the evaluation of the two utterances proceeded in terms of the three prosodic components. Of the three components, the evaluation of the F0 contour and the intensity contour was performed after the segmental durations of the two utterances were made identical. For this to work, the two utterances were manually aligned by the segments and their segmental durations were rendered identical using the technique proposed in [3]. For the evaluation of the segmental durations, however, no durational modifications were made to the two utterances and the differences between matching segmental durations were compared.

The durational modifications prior to the evaluation of the F0 and intensity contour was the key. Without any control in the durations of the component segments of the utterances involved in the evaluation, it would not be easy and fair to do the comparison. Since the two utterances are assumed to be composed of the same segments, once the component segments are made identical, the comparison can proceed only in the dimension of either the F0 contour or the intensity contour.

The evaluation was done separately in terms of each of the three prosodic components. That is, the evaluation in terms of the segmental durations is performed before any durational modifications are made. After the durational manipulation, the evaluation of the F0 contour is performed, followed by the evaluation of the intensity contour. Thus, the exaggerated utterance will have three separate evaluation scores with respect to the original non-exaggerated version. Since no experiments were performed with human evaluators, no suggestions can be made at this time as to how these separate scores can be accommodated to yield the overall score. The weights human evaluators would assign to each of the three evaluation scores could be assessed in the future work. In the current work, the exaggerated utterance will only have three separate scores.

The measure of differences used in the evaluation was the Euclidean distance metric between two matching F0 or intensity points. The same measure was used for the segmental durations. The Euclidean distance between two sets of points $P = (p1, p2, p3,..., pn)$ and $Q = (q1, q2, q3,..., qn)$ in Euclidean n-space is defined as follows. Here, the point P represents, for example, the pitch point values of the original utterance and the point Q represents those of the exaggerated utterance. As the equation shows, the two F0 contours, for example, are not seen as two separate curved lines in a two-dimensional plane, they are mathematically considered as two separate points in an n-dimensional space.

$$\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2}=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

Since the F0 range for the two speakers are different, normalization of the exaggerated version was performed prior to the calculation of the distance metric. The difference in the mean F0 values of the two utterances was added (or subtracted) to (or from) the F0 contour of the exaggerated version. With the two F0 contours in the normalized pitch range, a point-to-point comparison was made between matching pitch points (See <Figure 7>). The pitch points with no matching counterparts are not considered in the evaluation of the F0 contour. The farther away the exaggerated pitch points are from the original pitch points, the greater the Euclidean distance metric of all the pitch points considered.

Figure 7. The F0 contours of the original (Upper panel) and the exaggerated utterances (Lower panel). Since the segmental durations are the same, comparison of the matching pitch points can be made.

The same procedure can be applied in the evaluation of the intensity contours (See <Figure 8>). The normalization of the exaggerated utterance is done with respect to the difference in the overall intensity means of the two utterances. The normalization

is followed by the point-to-point comparison of the matching intensity points. The farther away the exaggerated intensity points are from the original intensity points, the greater the Euclidean distance metric among the intensity points involved.

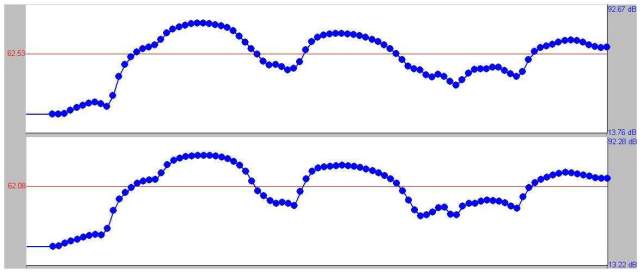

Figure 8. The intensity contours of the original (Upper panel) and the exaggerated utterances (Lower panel). Since the segmental durations are the same, comparison of the matching intensity points can be made.

The evaluation of the differences in the component segmental durations was performed without prior durational modifications. The differences in duration of the matching segments were used to calculate the Euclidean distance (See <Figure 9>). Since the utterances are labeled by the segments, one only needs to compare the segments in the label files. Automatic segmentation could help automate the labeling process.

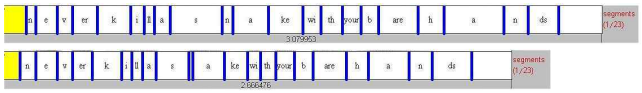

Figure 9. The durational differences between the two utterances of *Never kill a snake with your bare hands*.

2.4 An experiment

In order to examine the validity of using the Euclidean distance metric in the evaluation of each of the three prosodic components involved in the exaggeration, two utterances were produced. The same sentence was uttered by a native speaker of English (henceforth, the native) and by a native speaker of Korean (henceforth, the learner). The segmental compositions were the same and the two utterances were aligned and labeled as shown in <Figure 9> above. The learner utterance was given all the prosodic features of the native utterance by using the technique proposed in [3]. Now the two utterances are considered to have the same F0 contour, segmental durations and the intensity contour.

The next step was to exaggerate the learner utterance incrementally in terms of each of the three prosodic features. This was done using the Praat script given in the Appendix. As shown in <Figure 10>, the learner utterance was exaggerated from -100Hz to +100Hz with a 10Hz interval. Given the F0 contour exaggeration algorithm above, the -100Hz exaggeration means that the pitch peaks of the learner F0 contour were made lower, not higher, by 100Hz and the pitch valleys were made higher, not lower, by 100Hz (The gray arrows). The negative values mean the "reverse" exaggeration, making the peaks lower and the valleys higher. The positive F0 values mean making peaks higher and valleys lower (The black arrows). The matching pitch points between the native utterance and the learner utterance were compared to yield the Euclidean distance for the F0 contour. The prediction here is that the more exaggerated or deviant the learner utterance is than the native utterance, the greater the Euclidean distance would become. In other words, greater Euclidean distances mean more exaggerated, thus worse utterances compared to the native utterance.

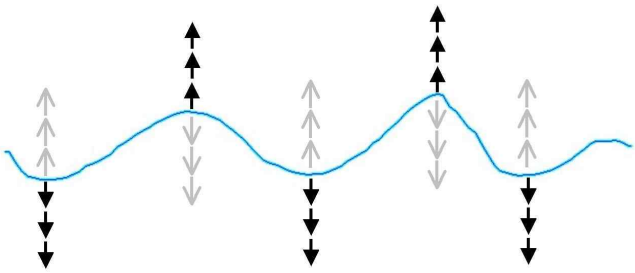

Figure 10. The creation of the experimental stimuli with different F0 contours. The black arrows indicate exaggerating with positive values and the gray arrows with negative values in the *freqValue* variable.

For the intensity contour, the learner utterance was exaggerated from -25dB to +25dB with a 5dB interval. The -25dB exaggeration means that the legitimate pitch peaks of the learner intensity contour were made lower, not higher, by 25dB. The negative intensity values mean the "reverse" exaggeration of the intensity contour, making the pitch peaks lower. The positive intensity values mean making the peaks higher in intensity. Note that nothing was done to the valleys. The matching intensity points between the native utterance and the learner utterance were compared to yield the Euclidean distance for the intensity contour. The same kind of prediction can be made. The more exaggerated, and thus deviant, the learner utterance is than the native utterance, the greater the Euclidean distance will be among the intensity points involved. Greater distance metrics mean worse utterances compared to the native utterance.

For the segmental durations, the pitch peaks of the learner utterance were exaggerated or lengthened 0.25, 0.50, 0.75, 1.00, 1.50,

2.00, 2.50, 3.00 times those of the original utterance. The values less than 1.00 mean that the segments were shortened. The values above 1.00 mean that they were lengthened. Note also that nothing was done to the valleys. Then the durations of the matching segments between the native utterance and the learner utterance were compared to yield the Euclidean distance for the durations. It can be predicted that more exaggerated utterances will yield greater Euclidean distances. Greater distance metrics mean worse utterances compared to the native utterance. The utterance used in the experiment was a sentence Never kill a snake with your bare hands read by a male native speaker of English (See <Figure 9>). From this template sentence, the durationally modified versions and all the exaggerated versions were synthesized.

3. Results

3.1 Prosody exaggeration

<Figure 11> shows the F0 contour of a one-word utterance *Marianna!* before and after the exaggeration of the F0 contour. The two pitch peaks were made higher and the valleys lower as defined in the F0 exaggeration algorithm. It also means that the two peaks were legitimate in the sense that the height of either slope of each peak was greater than the threshold frequency, which was 40Hz by default. The difference in frequency of the first peak between the two versions is 20Hz as specified in the variable freqValue.

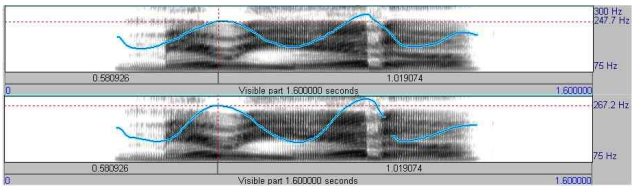

Figure 11. The exaggeration of the F0 contour of the utterance *Marianna!* before (Upper panel) and after (Lower panel) with the default value of 20Hz. Note that the pitch peaks became higher and the valleys lower. Compare the frequency values of the first peak (247Hz vs. 267Hz).

The overall shape of the two F0 contours is not exactly the same. The "loss" of the original F0 contour in the exaggerated version can be attributed to the artifact of the setting (2 semitones by default) of the stylization procedure in Praat. The *stylizeSemitones* variable can be changed to different values, but the default value works fine in most cases.

<Figure 12> shows the same utterance exaggerated in terms of the segmental durations. As defined by the duration exaggeration algorithm, only the legitimate pitch peaks were lengthened by 1.5 times as specified in the *durValue* variable. As shown in <Figure 5>, the lengthening starts from the immediately neighboring valley to the following peak. Thus it is the peak that is 1.5 times longer. The durations of the other segments lying on the uphill from the valley to the peak get gradually greater until it is 1.5 times the original duration at the peak. For the segments lying on the downhill from the peak to the following valley, the process is reversed. Compare the total durations of the two utterances. The exaggerated version is 0.165sec longer than the original version.

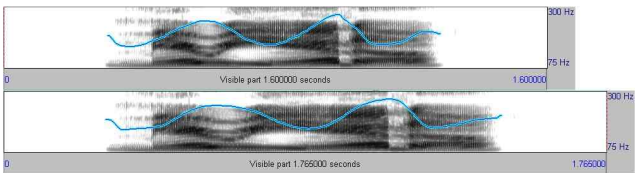

Figure 12. The exaggeration of the segmental durations of the utterance *Marianna!* before (Upper panel) and after (Lower panel) with the default value of 1.5 times. Note that only the legitimate pitch peaks were lengthened. Compare the total durations of the two utterances (1.600sec vs. 1.765).

<Figure 13> shows the intensity contour before and after the exaggeration. As in the duration exaggeration, the intensity exaggeration proceeded by gradually increasing the intensity values of the legitimate pitch peaks identified earlier in the process of exaggeration. Disregarding the F0 contour, it is clear that the intensity values in dB of the pitch peaks got greater in the exaggerated version. For the intensity values in the first peak (the dotted vertical line), it was around 76dB before and 83dB after the exaggeration.

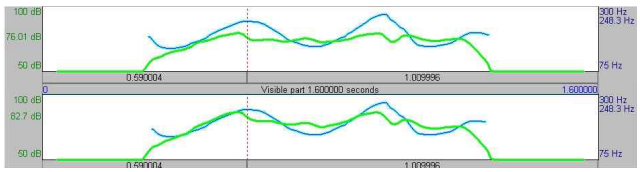

Figure 13. The exaggeration of the intensity contour of the utterance *Marianna!* before (Upper panel) and after (Lower panel) with the default value of 7dB for boosting peaks and 0.05sec intensity point interval. Only the legitimate peaks were made louder. Compare the intensity values at the first pitch peak (76.01dB vs.82.7dB).

<Figure 14> shows the result of performing exaggeration in terms of two or all three prosodic components. The upper panel is the original one-word utterance. The version in the middle panel was exaggerated in its F0 contour and segmental durations. The bottom panel added the intensity exaggeration. The

comparison of the F0 values, durations and intensity values at the first peak (the dotted vertical line) shows that the exaggeration was done as defined in the algorithm section. The F0 difference is around 20Hz (248.3Hz vs. 267.7Hz), the intensity value difference is around 7dB (76.14dB vs. 83.26dB) and the durational difference is 0.165sec.

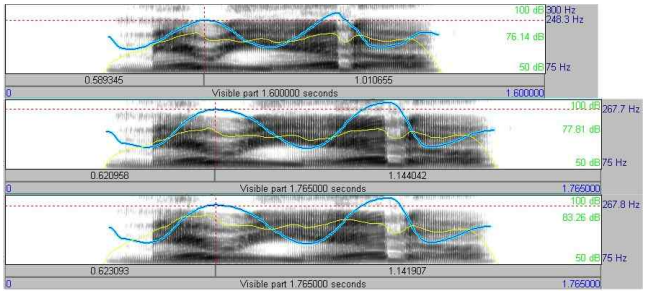

Figure 14. The exaggeration of the original utterance *Marianna!* (Upper panel) in terms of both the F0 contour and the segmental durations (Middle panel), and in terms of all the three prosodic components (Bottom panel). The cursor (the dotted vertical line) is at the first pitch peak.

3.2 Results of the experiment

The results from the experiment are shown in <Figure 15>, <Figure 16>, and <Figure 17>. <Figure 15> shows the Euclidean distance measures between the native utterance and the learner utterances with varying degrees of F0 exaggeration. The distance measure is minimum when no exaggeration is applied to the learner utterance (0 in the horizontal axis), in which case the learner utterance is the same as the native utterance in terms of the F0 contours. As the amount of exaggeration increases, so does the distance measure, making the overall plot "v"-shaped. This implies that the Euclidean distance measure can indeed be a good measure of evaluating the F0 contour differences. Note, however, that the measure is only valid among the utterances exaggerated following the scheme introduced in the algorithm section.

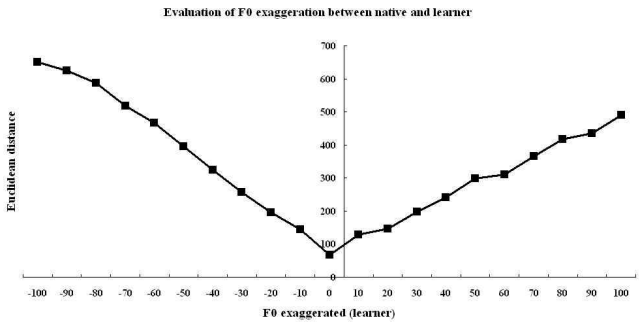

Figure 15. The evaluation of the learner utterances with exaggerated or deviant F0 contours. The Euclidean distance measures are plotted along the varying degrees of F0 exaggeration.

<Figure 16> shows the Euclidean distance measures between the native utterance and the learner utterances with varying degrees of intensity exaggeration. As in <Figure 15>, the overall pattern looks similar. The distance measure is minimum when no exaggeration is applied to the learner utterance. As the degree of the intensity exaggeration increases, so do the Euclidean distance measures, implying that the Euclidean distance can be a good measure for evaluating the intensity contour differences.

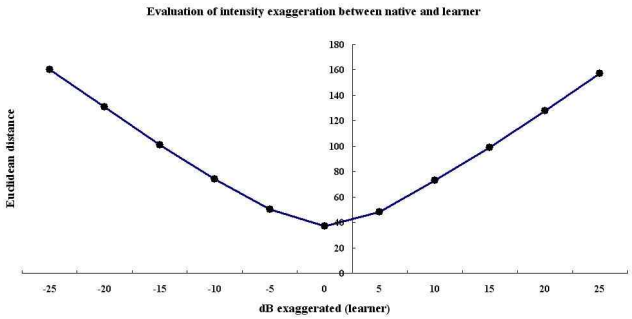

Figure 16. The evaluation of the learner utterances with exaggerated or deviant intensity contours. The Euclidean distance measures are plotted along the varying degrees of intensity exaggeration.

<Figure 17> shows the Euclidean distance measures between the native utterance and the learner utterances with varying degrees of duration exaggeration. Noting that a similar pattern can be seen here, we can also say that the Euclidean distance can work for evaluating the differences in segmental durations.

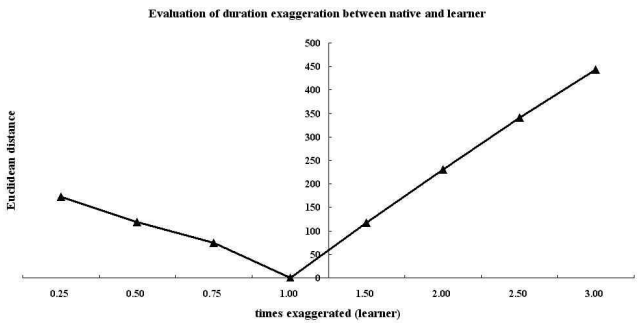

Figure 17. The evaluation of the learner utterances with exaggerated or deviant segmental durations. The Euclidean distance measures are plotted along the varying degrees of duration exaggeration.

As mentioned before, it is not known how these three separate measures of evaluation cooperate to yield the final measure of difference between the native utterance and the learner utterance. Experiments with human evaluators would reveal the "weights" they may implicitly assign in the evaluation of the overall

difference in prosody.

4. Conclusion

This paper introduced the technique of exaggerating one or more of the prosodic features of an utterance and implemented it in a Praat script. The technique was also used in an experiment designed to test the viability of using the Euclidean distance metric in the evaluation of the utterances with different prosodic features. The results from the experiment suggest that the Euclidean distance metric can be a good measure in evaluating utterances in terms of each of the three prosodic features. However, no suggestions can be made at this time as to these separate measures are combined to yield the overall prosodic impression of an utterance. Experiments with human evaluators could reveal the interactions of the measures at the perceptual level.

The technique of manipulating one or more of the three prosodic features of an utterance can be applied in many areas of study. For example, it can be used in a perception experiment designed to examine the role and the interaction of each of the three prosodic features in sentence processing. In English education, this technique can be employed to develop a tool for teaching the prosody of the target language and to help improve the pronunciation capability of the students at the sentence level as well as at the word level.

The use of the Euclidean distance metric in the evaluation of the utterances with different prosodic features needs to be verified with more experiments. The experimental stimuli should cover more utterances with prosodic features different from the ones used in the current study. When the interactions of the three separate evaluation measures are discovered, the ultimate measure could be used in the automatic evaluation of the utterance prosody.

One limitation of the current work was that the prosody of the learner utterance was evaluated with respect to that of the native utterance of the same text. In a more advanced situation, any utterances of the learner should be evaluated without their native speaker versions. With a perfect prosody model of the target language, the task could be achieved. A more practical alternative could be to get help from the technology of automatic text-to-speech (TTS) synthesis systems with more advanced prosody models. For any utterances produced by the learner, the TTS system could provide their native versions so that the comparisons can be made to yield the evaluation measures.

References

[1] Boersma, P. (2001). “Praat, a system for doing phonetics by computer”, *Glot International,* Vol. 5, No. 9/10, pp. 341-345.

[2] Moulines, E. & Charpentier, F. (1990). “Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones”, *Speech Communication*, Vol. 9, pp. 453-467.

[3] Yoon, K. (2007). “Imposing native speakers' prosody on non-native speakers' utterances: The technique of cloning prosody”, *Journal of the Modern British & American Language & Literature,* Vol. 25, No. 4, pp. 197-215.

• **윤규철 (Yoon, Kyuchul)**
영남대학교 영어영문학부
경상북도 경산시 대동 214-1
Tel: 053-810-2145 Fax: 053-810-4607
Email: kyoon@ynu.ac.kr
관심분야: 음성학, 음운론
현재 영어영문학부 교수

Appendix

1. Script for prosody exaggeration

```
############################################################
# Given a sound file, this script stylizes the original pitch contour,
# and exaggerates its F0 and intensity contour, and peak durations
# automatically.
# Version 3 ==> Either slope of a peak needs to be higher than the
# threshold value.
# NOTE) Users can check graphically how the original sound gets
#           exaggerated by following the pop-up Manipulation Editor
#           window.
# NOTE) Users can also select which parameter(s) to exaggerate:
#           (1) F0 contour only
#           (2) Duration only (for the F0 peaks)
#           (3) Intensity contour only (for the F0 peaks)
#           (4) F0 + Duration only                          (1)+(2)
#           (5) F0 + Intensity contour only                 (1)+(3)
#           (6) Duration + Intensity contour only           (2)+(3)
#           (7) All: F0 + Duration + Intensity              (1)+(2)+(3)
############################################################
# F0 exaggeration algorithm:
# 1. Compare two neighboring pitch points in Hz.
# 2. The point with higher Hz gets boosted, while the lower one gets
#    lowered. If same, nothing happens. (Check version 3 changes above)
# 3. Move to the next two points, the first of which is the second
#    from the previous step. (i.e. advances by one pitch point)
#    Use values before boosting/lowering from the previous step.
# 4. Repeat step 1.
####################################################################
# Duration exaggeration algorithm:
# 1. Lengthen stylized pitch tier object peaks durValue times.
# 2. Neighboring valleys maintain the original duration.
####################################################################
# Intensity exaggeration algorithm:
# 1. Given a peak, the exaggeration begins from neighboring valleys;
#    If the user specifies 10dB for example, and the intervening
#    intensity points are ten between the valley and the peak, the dB
#    value for each point is incrementally increased. The first point,
#    corresponding to the valley point, is increased by 1dB, the second
#    by 2dB,..., the tenth, corresponding to the peak, by 10dB. The
#    same thing repeats for the downhill (from peak to valley) slope.
####################################################################
# Script installation procedure:
# (1) Start Praat : Praat > Open Praat script...
# (2) Open this script "exaggerateF0-duration-button.praat
# (3) In the script editor window : File > Add to dynamic menu...
# (4) Replace "Do it..." with something like "Exaggerate!"
# (5) Click [OK]
# (6) Close Script Editor window
# (7) You can now launch the script from within Praat using the
#     "Exaggerate!" button when a sound object is selected.
####################################################################
form Parameters
  comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
  comment F0 CONTOUR EXAGGERATION by boosting peaks & lowering valleys
  comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
  choice ExaggerateFrequency: 1
    button Yes
    button No
      natural stylizeSemitones 2
      natural freqValue_(Hz_diff_for_boosting_and_lowering) 20
      comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||
      comment DURATIONS EXAGGERATION by lengthening peaks
      comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||
      choice ExaggerateDuration: 1
            button Yes
            button No
      real durValue_(times_orig_duration_for_lengthening) 1.5
      comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||
      comment INTENSITY CONTOUR EXAGGERATION
      comment ||||||||||||||||||||||||||||||||||||||||||||||||||||||
      choice ExaggerateIntensity: 1
            button Yes
            button No
      real intTimeStep 0.05
      real dBValue_(dB_boosting_for_peaks) 7
      comment ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      comment For peak selection
      natural freqThreshold_(peak_height_in_Hz) 40
      comment ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      comment For resynthesis (Default value would suffice)
      natural numFreqPoints_(per_parabola_for_resynthesis) 8
endform

# Create a manipulation object for the sound object and stylized it.
Copy... soundObj
To Manipulation... 0.01 75 600
Rename... inFileManipObj
Copy... original
select Manipulation inFileManipObj
Edit
editor Manipulation inFileManipObj
            Stylize pitch... stylizeSemitones Semitones
            Close
endeditor
Copy... original_stylized

# Create and store a pitch object from the manipulation object.
select Manipulation inFileManipObj
Extract pitch tier
Rename... inFilePitchTierObj
numPitchPoints = Get number of points

# Flags to be used later on.
flagIdentifyPeakOnce = 0
flagUseExaggeratedF0 = 0
flagUseExaggeratedDur = 0

#####################################################
# Check user's choice and call appropriate procedures #
# (1) F0 contour only
if (exaggerateFrequency$ = "Yes" and exaggerateDuration$ = "No"
            ...and exaggerateIntensity$ = "No")
            call procExaggerateFreq freqValue
# (2) Duration only (for the F0 peaks)
elsif (exaggerateFrequency$ = "No" and exaggerateDuration$ = "Yes"
            ...and exaggerateIntensity$ = "No")
            call procExaggerateDur durValue
# (3) Intensity contour only (for the F0 peaks)
elsif (exaggerateFrequency$ = "No" and exaggerateDuration$ = "No"
            ...and exaggerateIntensity$ = "Yes")
            call procExaggerateInt dBValue
# (4) F0 + Duration only: (1)+(2)
elsif (exaggerateFrequency$ = "Yes" and exaggerateDuration$ = "Yes"
            ...and exaggerateIntensity$ = "No")
            call procExaggerateFreq freqValue
            call procExaggerateDur durValue
# (5) F0 + Intensity contour only: (1)+(3)
elsif (exaggerateFrequency$ = "Yes" and exaggerateDuration$ = "No"
            ...and exaggerateIntensity$ = "Yes")
            # If flag is 1, then the procExaggerateInt must use
            # "exaggeratedF0" sound from earlier procedures.
            flagUseExaggeratedF0 = 1
            call procExaggerateFreq freqValue
            call procExaggerateInt dBValue
# (6) Duration + Intensity contour only: (2)+(3)
elsif (exaggerateFrequency$ = "No" and exaggerateDuration$ = "Yes"
            ...and exaggerateIntensity$ = "Yes")
            # If flag is 1, then no need to identify the peak locations
            # twice.
            flagIdentifyPeakOnce = 1
            # If flag is 1, then the procExaggerateInt must use
            # "exaggeratedDur" sound from earlier procedures.
            flagUseExaggeratedDur = 1
            call procExaggerateDur durValue
            call procExaggerateInt dBValue
# (7) All: F0 + Duration + Intensity: (1)+(2)+(3)
elsif (exaggerateFrequency$ = "Yes" and exaggerateDuration$ = "Yes"
            ...and exaggerateIntensity$ = "Yes")
            # If flag is 1, then no need to identify the peak
            # locations twice.
            flagIdentifyPeakOnce = 1
            # If flag is 1, then the procExaggerateInt must use
            # "exaggeratedDur" sound from earlier procedures.
            flagUseExaggeratedDur = 1
            call procExaggerateFreq freqValue
            call procExaggerateDur durValue
            call procExaggerateInt dBValue
# User selected three No's.
else
            exit All I need is one or more Yes!
endif
############### END OF MAIN FUNCTION ###############

#######################################
# Procedure for exaggerating F0 contour #
procedure procExaggerateFreq fValue
  # 'Exaggerating' loop. The exaggeration is performed on the pitch
  # tier object
  select PitchTier inFilePitchTierObj

  # A flag to signal whether a pitch point was modified or not
  flagModified = 0
  for i to numPitchPoints
    # For all the pitch points except for the last one.
  # This is necessary because the 'exaggerating' is done for two
    # neighboring pitch points. The last one has no such neighbor.
    if i <> numPitchPoints
      # Special treatment for the first pitch point
      if flagModified = 0
            iPrevHz = Get value at index... i
            # If it's not the first pitch point, the next pitch point
            # from the earlier loop becomes the previous pitch point
            # in the current loop
            else
                        iPrevHz = iNextHz
            endif

            # Get the Hz value and the time values of the two points
            iNextHz = Get value at index... (i+1)
            iPrevTime = Get time from index... i
            iNextTime = Get time from index... (i+1)
            freqDiff = abs(iNextHz-iPrevHz)

            # Proceed if the frequency difference is bigger than the
            # threshold
            if freqDiff > freqThreshold
              # Decide which is bigger
              biggerHz = imax(iPrevHz, iNextHz)
              # If the first of the two pitch points is bigger, boost it
```

```
            # by freqValue and lower the second by as much
            if biggerHz = 1
                      flagModified = 1
                      Remove point... i
                      Add point... iPrevTime (iPrevHz+fValue)
                      Remove point... (i+1)
                      Add point... iNextTime (iNextHz-fValue)
            # If the second is bigger, then do similar jobs
            elsif biggerHz = 2
                      flagModified = 1
                      Remove point... i
                      Add point... iPrevTime (iPrevHz-fValue)
                      Remove point... (i+1)
                      Add point... iNextTime (iNextHz+fValue)
            # If same, do nothing
            else
                      # Do nothing
                      flagModified = 0
            endif
          endif
  endif
endfor

  # Switch the exaggerated pitch tier object
  plus Manipulation inFileManipObj
  Replace pitch tier
  select Manipulation inFileManipObj
  Edit
  editor Manipulation inFileManipObj
    Interpolate quadratically... numFreqPoints
    Publish resynthesis
    Close
  endeditor
  Rename... exaggerated_F0
endproc

########################################
# Procedure for exaggerating durations #
procedure procExaggerateDur dValue
  # To exaggerate durations, the locations of peaks and valleys
  # need to be identified.
  call procIdentifyPeaks
  select Manipulation inFileManipObj
  Extract duration tier
  Rename... inFileDurTierObj
  # Add duration points at the peak locations
  for m to k
            peakTime = timeOfPeaks'm'
            leftValleyTime = timeOfLeftValley'm'
            rightValleyTime = timeOfRightValley'm'
            Add point... peakTime dValue
            Add point... leftValleyTime 1
            Add point... rightValleyTime 1
  endfor

  # Switch the exaggerated duration tier object
  plus Manipulation inFileManipObj
  Replace duration tier
  select Manipulation inFileManipObj
  Edit
  editor Manipulation inFileManipObj
            Interpolate quadratically... numFreqPoints
            Publish resynthesis
            Close
  endeditor
  Rename... exaggerated_Dur
endproc

################################################
# Procedure for exaggerating intensity contour #
procedure procExaggerateInt iValue
  # To exaggerate the intensity, the locations of peaks and valleys
  # need to be identified. If the flag is 1, then we already got
  # the locations
  # from procExaggerateDur procedure.
  if flagIdentifyPeakOnce = 0
            call procIdentifyPeaks
  endif

  if flagUseExaggeratedDur = 1
            select Sound exaggerated_Dur
  elsif flagUseExaggeratedF0 = 1
            select Sound exaggerated_F0
  else
            select Sound soundObj
  endif

  To Intensity... 100 intTimeStep Yes
  Rename... intensityObj
  Copy... intensityCopy
  select Intensity intensityObj
  Down to IntensityTier
  Rename... intensityTierObj
  Copy... intensityTierCopy
  select IntensityTier intensityTierObj

  # For each peak and its surrounding valleys, exaggerate
  # intensity contour.
  for s to k
    peakTime = timeOfPeaks's'
    leftValleyTime = timeOfLeftValley's'
    rightValleyTime = timeOfRightValley's'

    # Exaggerate the uphill to the peak. Count the number of
    # intensity points
    select IntensityTier intensityTierObj
    indexOfLtValley = Get low index from time... leftValleyTime
    indexOfPeak = Get low index from time... peakTime
    numOfIntensityPoints = (indexOfPeak - indexOfLtValley) + 1
    # Determine the increment for each intensity point.
    pointIncrement = dBValue / numOfIntensityPoints
    # Now, rearrange the intensity points
            for u to numOfIntensityPoints
              realIndexOfPoint = (indexOfLtValley + u) - 1
              select IntensityTier intensityTierObj
              timeOfPoint = Get time from index... realIndexOfPoint

              select Intensity intensityObj
              oldIntensityValue = Get value at time... timeOfPoint Cubic
              newIntensityValue = oldIntensityValue +
                        ...(pointIncrement * u)

              select IntensityTier intensityTierObj
              Remove point... realIndexOfPoint
              Add point... timeOfPoint newIntensityValue
            endfor

    # Exaggerate the downhill to the peak. Count the number of
    # intensity points
    select IntensityTier intensityTierObj
    # We already have the indexOfPeak from the uphill codes.
    indexOfRtValley = Get low index from time... rightValleyTime
    numOfIntensityPoints = (indexOfRtValley - indexOfPeak) + 1
    # Determine the increment for each intensity point.
    pointIncrement = dBValue / numOfIntensityPoints
    # Now, rearrange the intensity points
    for v to numOfIntensityPoints
            x = (numOfIntensityPoints + 1) - v
            realIndexOfPoint = (indexOfPeak + v) - 1
            select IntensityTier intensityTierObj
            timeOfPoint = Get time from index... realIndexOfPoint

            select Intensity intensityObj
            oldIntensityValue = Get value at time... timeOfPoint Cubic
            newIntensityValue = oldIntensityValue + (pointIncrement * x)

            select IntensityTier intensityTierObj
            Remove point... realIndexOfPoint
            Add point... timeOfPoint newIntensityValue
    endfor
  endfor

  # Before switching the exaggerated intensityTier object, neutralize
  # the intensityTier object of the sound object first.
  select Intensity intensityObj
  # Inverse the intensity object by getting the maximum and
  # subtracting self
  maxInt = Get maximum... 0 0 Parabolic
  Formula... 'maxInt' - self
  # And make IntensityTier object
  Down to IntensityTier
  Rename... inverseIntensityTierObj

  # Multiply the sound with its own inverse IntensityTier.
  if flagUseExaggeratedDur = 1
            select Sound exaggerated_Dur
  elsif flagUseExaggeratedF0 = 1
            select Sound exaggerated_F0
  else
            select Sound soundObj
  endif
  plus IntensityTier inverseIntensityTierObj
  Multiply
  Rename... neutralizedSoundObj
  # And then by the exaggerated IntensityTier.
  plus IntensityTier intensityTierObj
  Multiply
  Rename... exaggerated_Int
endproc

############################################
# Procedure for identifying peak locations #
procedure procIdentifyPeaks
  select PitchTier inFilePitchTierObj
  # Identify the pitch peaks. Initialize the index of the peak
  # array variable
  k = 0
  # Except for the first and last pitch point, check if it's a peak
  for j from 2 to (numPitchPoints-1)
    leftPointFreq = Get value at index... (j-1)
    rightPointFreq = Get value at index... (j+1)
    currentPointFreq = Get value at index... j
    indexOfMax = imax(leftPointFreq, currentPointFreq, rightPointFreq)

    # Values to be checked to see if the peak is high enough
    leftHeight = abs(currentPointFreq-leftPointFreq)
    rightHeight = abs(currentPointFreq-rightPointFreq)

    # If the current point is at the peak, remember the time coordinate
    # in an array variable timeOfPeaks. Remember its neighboring
    # valleys
    if indexOfMax = 2
            if (leftHeight > freqThreshold or rightHeight >
            ...freqThreshold)
              k = k + 1
```

```
            timeOfPeaks'k' = Get time from index... j
            timeOfLeftValley'k' = Get time from index... (j-1)
            timeOfRightValley'k' = Get time from index... (j+1)
          endif
   endif
  endfor
endproc
############## END OF SCRIPT ################

2. Script for F0 contour evaluation
#####################################################################
# Evaluates a set of native/learner sound files in separate folders in
# terms of their F0 contours. They should have .TextGrid files labeled
# by segments. The number of segments should be the same. Another
# script "prosodyCloning-durationOnly.praat" should be in the same
# folder as this one.
#####################################################################
# -------------------- F0 evaluation algorithm --------------------
# 1. Make the two sound files have the same segmental durations.
# 2. Normalize the learner F0 contour with respect to the native F0.
# 3. Extract the pitch contours and make them smooth.
# 4. Do the point-to-point comparison and log the Hz differences.
#####################################################################
# For 1, use the prosody-cloning script written by Kyuchul Yoon and
#           get the duration-modified version from the result files,
#           i.e. the new learner file should have the same segmental
#           durations as those of the native file. Use the new one
#           from now on.
# For 2, query the mean pitch values from the two Pitch objects and
#           add/subtract the difference to/from the learner F0 contour
#           by using the "shift pitch frequencies" command from the
#           Manipulation Editor window. Republish synthesis. Now, we
#           have a new normalized learner sound file to use.
# For 3, using the two updated versions, extract their Pitch objects
#           and smooth them using the "Smooth..." command of the Pitch
#           objects.
# For 4, using the two smoothed Pitch objects, do a point-to-point
#           comparison and log the Hz differences.
#####################################################################
form Specify parameters
  word nativeFolder_(with_native_utterances) native
  word nativeSound_(with_dot_wav) native.wav
  word nativeTextgrid_(with_dot_TextGrid) native.TextGrid
  word learnerFolder_(with_learner_utterances) learner
  word learnerSound_(with_dot_wav) ky_NeverKillASnake.wav
  word learnerTextgrid_(with_dot_TextGrid) ky_NeverKillASnake.TextGrid
  natural tierNumber 1
  comment A log file will be created with the filename prefix.
endform

prefix$ = learnerSound$ - ".wav"
logFile$ = prefix$ + ".log"
# Print the header line for the log file.
fileappend 'logFile$' native'tab$'learner'tab$'numFrames'tab$'
           ...frameNo'tab$'time'tab$'nativeF0'tab$'learnerF0'tab$'
           ...diffF0'newline$'

# For 1, use another script. This will create a duration-modified
# version of the learner file. The sound object name is
# synNonnatSoundObjD.
execute prosodyCloning-durationOnly.praat 'nativeFolder$'
           ...'nativeSound$' 'nativeTextgrid$' 'tierNumber'
           ...'learnerFolder$' 'learnerSound$' 'learnerTextgrid$'
           ...'tierNumber' 'learnerFolder$'

# For 2, compare the mean of the overall pitch values and do the
# normalization.
Read from file... 'nativeFolder$'/'nativeSound$'
Rename... nativeSoundObj
To Pitch... 0 75 600
Rename... nativePitchObj
nativeMeanHz = Get mean... 0 0 Hertz
# Get the smoothed version for later pitch point comparison.
Smooth... 10
Rename... smoothNativePitchObj

select Sound synNonnatSoundObjD
Rename... learnerSoundObj
To Pitch... 0 75 600
Rename... learnerPitchObj
learnerMeanHz = Get mean... 0 0 Hertz
meanHzDiff = nativeMeanHz - learnerMeanHz

# Now that we got the mean F0 difference, add/subtract it in the
# manipulation editor.
select Sound learnerSoundObj
totalDuration = Get total duration
To Manipulation... 0.01 75 600
Rename... learnerManipObj
Edit
editor Manipulation learnerManipObj
           Select... 0 totalDuration
           Shift pitch frequencies... meanHzDiff Hertz
           Publish resynthesis
           Close
endeditor
# Now we have the normalized (wrt/ mean Hz difference) version of the
# learner sound.
Rename... normLearnerSoundObj
# Save the file in the learner folder.
normLearnerSound$ = "normalized-" + learnerSound$
Write to WAV file... 'learnerFolder$'/'normLearnerSound$'

# For 3, use the the duration-modified F0 normalized version of the
# learner file to get the final smoothed pitch object.
To Pitch... 0 75 600
Smooth... 10
Rename... smoothLearnerPitchObj

# For 4, use the two Pitch objects because they have the same number
# of frames. Do a frame-to-frame comparison. For each frame, query
# the time and get the Hz from the two PitchTier objects.
select Pitch smoothNativePitchObj
numFrames = Get number of frames
# Loop through each frame
iCount = 0
for iFrame to numFrames
           select Pitch smoothNativePitchObj
           timeOfFrame = Get time from frame number... iFrame
           nativeF0 = Get value in frame... iFrame Hertz
           select Pitch smoothLearnerPitchObj
           learnerF0 = Get value in frame... iFrame Hertz
           diffF0 = nativeF0 - learnerF0

           # If diffF0 is not undefined, store the values in an array
           # variable.
           if diffF0 <> undefined
                      iCount = iCount + 1
                      arrayDiffF0'iCount' = diffF0
           endif
           # Save the information in the log file.
           fileappend 'logFile$' 'nativeSound$''tab$''learnerSound$'
           ...'tab$''numFrames''tab$''iFrame''tab$''timeOfFrame:3'
           ...'tab$''nativeF0:0''tab$''learnerF0:0''tab$'
           ...'diffF0:0''newline$'
endfor

# Calculate the sums of squares
sum = 0
for i to iCount
           dummy = arrayDiffF0'i'
           squareValue = dummy * dummy
           sum = sum + squareValue
endfor
squareRootSum = sqrt(sum)
fileappend 'logFile$' 'newline$''newline$'Sums of squares of diffF0's
           ...is 'sum:0'
fileappend 'logFile$' 'newline$''newline$'Square root of the sums is
           ...'squareRootSum:0'
select all
#Remove
################### END OF SCRIPT ###################

3. Script for intensity evaluation
#####################################################################
# Evaluates a set of native/learner sound files in separate folders
# in terms of their intensity contour. They should have .TextGrid
# files labeled by segments. The number of segments should be the
# same. Another script "prosodyCloning-durationOnly.praat" should be
# in the same folder as  this one.
#####################################################################
# ------------- Intensity evaluation algorithm -------------------
# 1. Make the two sound files have the same segmental durations.
# 2. Normalize the learner intensity contour with respect to the
#           native intensity.
# 3. Extract the intensity contours.
# 4. Do the point-to-point comparison and log the dB differences.
#####################################################################
# For 1, use the prosody-cloning script written by Kyuchul Yoon and
#         get the duration-modified version from the result files, i.e.
#         the new learner file should have the same segmental durations
#         as those of the native file. Use the new one from now on.
# For 2, query the mean dB values from the two Intensity objects and
#           add/subtract the difference to/from the learner intensity
#           contour by using the "Formula..." command objects window.
#           Now, we have a new normalized learner sound file to use.
# For 3, using the two updated versions, extract their Intensity
#           objects.
# For 4, using the two Intensity objects, do a point-to-point
#           comparison and log the dB differences.
#####################################################################
form Specify parameters
  word nativeFolder_(with_native_utterances) native
  word nativeSound_(with_dot_wav) NeverKillASnake.wav
  word nativeTextgrid_(with_dot_TextGrid) NeverKillASnake.TextGrid
  word learnerFolder_(with_learner_utterances) learner
  word learnerSound_(with_dot_wav) ky_NeverKillASnake.wav
  word learnerTextgrid_(with_dot_TextGrid) ky_NeverKillASnake.TextGrid
  natural tierNumber 1
  comment A log file will be created with the filename prefix.
endform

prefix$ = learnerSound$ - ".wav"
logFile$ = prefix$ + "-dB.log"
# Print the header line for the log file.
fileappend 'logFile$' native'tab$'learner'tab$'numFrames'tab$'frameNo
...'tab$'time'tab$'nativedB'tab$'learnerdB'tab$'diffdB'newline$'

# For 1, use another script. This will create a duration-modified
# version of the learner file. The sound object name is
# synNonnatSoundObjD.
execute prosodyCloning-durationOnly.praat
           ...'nativeFolder$''nativeSound$'
           ... 'nativeTextgrid$' 'tierNumber' 'learnerFolder$'
           ...'learnerSound$' 'learnerTextgrid$' 'tierNumber'
```

```
            ... 'learnerFolder$'

# For 2, compare the mean of the overall intensity values and do the
# normalization.
Read from file... 'nativeFolder$'/'nativeSound$'
Rename... nativeSoundObj
To Intensity... 100 0 yes
Rename... nativeIntensityObj
nativeMeandB = Get mean... 0 0 dB

select Sound synNonnatSoundObjD
Rename... learnerSoundObj
To Intensity... 100 0 yes
Rename... learnerIntensityObj
learnerMeandB = Get mean... 0 0 dB
meandBDiff = nativeMeandB - learnerMeandB

# Now that we got the mean dB difference, add/subtract it in the
# Intensity object.
select Intensity learnerIntensityObj
Formula... self + meandBDiff
Rename... normLearnerIntensityObj

# For 3, we now have the normalized (wrt/ mean dB difference) version
# of the learner Intensity Object. Save it to an intensity file.
Rename... normLearnerIntensityObj
# Save the file in the learner folder.
normLearnerIntensity$ = "normalized-" + prefix$
Write to text file... 'learnerFolder$'/'normLearnerIntensity$'

# For 4, use the two Intensity objects because they have the same
# number of frames. Do a frame-to-frame comparison. For each frame,
# query the time and get the dB from the two Intensity objects.
select Intensity nativeIntensityObj
numFrames = Get number of frames
# Loop through each frame
iCount = 0
for iFrame to numFrames
            select Intensity nativeIntensityObj
            timeOfFrame = Get time from frame number... iFrame
            nativedB = Get value in frame... iFrame
            select Intensity normLearnerIntensityObj
            learnerdB = Get value in frame... iFrame
            diffdB = nativedB - learnerdB

            # If diffdB is not undefined, store the values in an array
            # variable.
            if diffdB <> undefined
                        iCount = iCount + 1
                        arrayDiffdB'iCount' = diffdB
            endif

            # Save the information in the log file.
            fileappend 'logFile$' 'nativeSound$''tab$''learnerSound$'
                        ...'tab$''numFrames''tab$''iFrame''tab$'

...'timeOfFrame:3''tab$''nativedB:2''tab$''learnerdB:2'
                        ...'tab$''diffdB:2''newline$'
endfor

# Calculate the sums of squares
sum = 0
for i to iCount
            dummy = arrayDiffdB'i'
            squareValue = dummy * dummy
            sum = sum + squareValue
endfor
squareRootSum = sqrt(sum)
fileappend 'logFile$' 'newline$''newline$'Sums of squares of diffdB's
            ...is 'sum:0'
fileappend 'logFile$' 'newline$''newline$'Square root of the sums
            ...is 'squareRootSum:0'
select all
Remove
################### END OF SCRIPT ###################

4. Script for duration evaluation
#####################################################################
# Evaluates a set of native/learner sound files in separate folders
# in terms of their segmental durations. They should have .TextGrid
# files labeled by segments. The number of segments should be the same.
#####################################################################
# ---------------- Duration evaluation algorithm --------------------
# 1. The native/learner utterances should be labeled segmentally
#           (depending on your definition of segments)
# 2. Do a segment-by-segment comparison between the two and log the
#           duration differences.
#####################################################################
form Specify parameters
  word nativeFolder_(with_native_utterances) native
  word nativeTextgrid_(with_dot_TextGrid) NeverKillASnake.TextGrid
  word learnerFolder_(with_learner_utterances) learner
  word learnerTextgrid_(with_dot_TextGrid) ky_NeverKillASnake.TextGrid
  natural tierNumber 1
  comment A log file will be created with the filename prefix.
endform

prefix$ = learnerTextgrid$ - ".wav"
logFile$ = prefix$ + "-msec.log"
# Print the header line for the log file.
fileappend 'logFile$'
native'tab$'learner'tab$'numSegs'tab$'segNo'tab$'
            ...nativeSegID'tab$'learnerSegID'tab$'timeStart'tab$'
            ...nativeDur'tab$'learnerDur'tab$'diffDur'newline$'

# Since all the sounds to be compared were labeled, no need to load
# the sounds. Just load the .TextGrid files.
Read from file... 'nativeFolder$'/'nativeTextgrid$'
Rename... nativeObj
Read from file... 'learnerFolder$'/'learnerTextgrid$'
Rename... learnerObj

select TextGrid nativeObj
nativeNumIntervals = Get number of intervals... tierNumber
select TextGrid learnerObj
learnerNumIntervals = Get number of intervals... tierNumber
# Check if the number of segments is the same, otherwise exit.
if nativeNumIntervals <> learnerNumIntervals
            exit Warning! The number of segments do not match!
endif

# Loop through each interval (or segment) for duration comparison.
for iInterval to nativeNumIntervals
  # For the native TextGrid object.
  select TextGrid nativeObj
  # Get the label of the current segment.
  nativeLabOfInterval$ = Get label of interval... tierNumber iInterval
  # If there is no label in the current interval, then give it
  # "No_label".
  if length(nativeLabOfInterval$) = 0
            nativeLabOfInterval$ = "No_label"
  endif
  # Get the duration of the current segment.
  nativeStartOfInterval = Get starting point... tierNumber iInterval
  nativeEndOfInterval = Get end point... tierNumber iInterval
  nativeDurOfSeg = nativeEndOfInterval - nativeStartOfInterval
  nativeDur = nativeDurOfSeg * 1000

  # For the learner TextGrid object
  select TextGrid learnerObj
  # Get the label of the current segment.
  learnerLabOfInterval$ = Get label of interval... tierNumber iInterval
  # If there is no label in the current interval, then give it \
  # "No_label".
  if length(learnerLabOfInterval$) = 0
            learnerLabOfInterval$ = "No_label"
  endif
  # Get the duration of the current segment.
  learnerStartOfInterval = Get starting point... tierNumber iInterval
  learnerEndOfInterval = Get end point... tierNumber iInterval
  learnerDurOfSeg = learnerEndOfInterval - learnerStartOfInterval
  learnerDur = learnerDurOfSeg * 1000

  # Get the difference in duration of the two matching segments.
  diffDur = nativeDur - learnerDur
  arrayDiffDur'iInterval' = diffDur
  # Save the information in the log file.
  fileappend 'logFile$' 'nativeTextgrid$''tab$''learnerTextgrid$'
...'tab$''nativeNumIntervals''tab$''iInterval''tab$'
...'nativeLabOfInterval$''tab$''learnerLabOfInterval$'
...'tab$''nativeStartOfInterval:3''tab$''nativeDur:0''tab$'
...'learnerDur:0''tab$''diffDur:0''newline$'
endfor

# Calculate the sums of squares
sum = 0
for i to nativeNumIntervals
            dummy = arrayDiffDur'i'
            squareValue = dummy * dummy
            sum = sum + squareValue
endfor
squareRootSum = sqrt(sum)
fileappend 'logFile$' 'newline$''newline$'Sums of squares of diffDur's
            ...is 'sum:0'
fileappend 'logFile$' 'newline$''newline$'Square root of the sums is
            ...'squareRootSum:0'
select all
Remove
################### END OF SCRIPT ###################
```