CDP Spectral MORPHING

(with Command Line Usage)

Functions to Morph Transitions Between Sounds

(Names in brackets mean that these are separate programs. The others are sub-modules of MORPH.)

BRIDGE
Interpolate between a specified window in one file, and another window specified in another file

GLIDE
Interpolate between two single window spectra

MORPH
Morph between one spectrum and another, where spectra may be time-varying

[NEWMORPH]
Morph between dissimilar spectra

[NEWMORPH2]
Morph frequencies of spectral peaks

MORPH BRIDGE – make a bridging-interpolation between two sound spectra by interpolating between 2 time-specified windows in the 2 infiles

Usage

morph bridge mode *infile1 infile2 outfile* [**-a***start*] [**-b***end*] [**-c***sf2*] [**-d***ef2*] [**-e***sa2*] [**-f***ea2*]

Modes

1 Output level is the direct result of interpolation
2 Output level follows moment to moment minimum of the 2 *infile* amplitudes
3 Output level follows moment to moment amplitude of *infile1*
4 Output level follows moment to moment amplitude of *infile2*
5 Output level moves, through interpolation, from that of *infile1* to that of *infile2*
6 Output level moves, through interpolation, from that of *infile2* to that of *infile1*

Parameters

infile, infile2 – input analysis files made with PVOC
outfile – output analysis file
-a*start* time of startwindow for interpolation, in secs: Default: 0.0
-b*end* time of endwindow for interpolation, in secs: Default: end_of_file
-c*sf2* fraction of 2nd sound's frq interpolated at *start*; (Range: 0 to 1, Default 0)
-d*ef2* fraction of 2nd sound's frq interpolated at *end* (Default 1)
-e*sa2* fraction of 2nd sound's amp interpolated at *start* (Default 0)
-f*ea2* fraction of 2nd sound's amp interpolated at *end* (Default 1)

Understanding the MORPH BRIDGE Process

Formerly SPECINTE, this process interpolates between two existing soundfiles. Unlike MORPH MORPH (formerly VOCINTE), this process interpolates between the fixed state of one sound at a specific time in that soundfile, and the fixed state of the 2nd sound at a specific time in that 2nd soundfile. The interpolation only works smoothly with sounds of quite stable spectrum.

You can get an idea of what this means by understanding how the vibrato on a sound can be lost during a BRIDGE process. When a window (fixed state) is taken from the first soundfile, the data is as it were frozen at that point. Then an interpolation takes place simply between the data in the two windows, the two 'frozen' states in each file, not on the data in the first file after the window taken, or on the data in the second file before the window taken. Thus, if there were some vibrato on the first sound after the point at which the window is taken, it is lost.

For more unstable, or noisier sounds, use MORPH MORPH, because this actually works on the data *within* the time period selected.

Note that there is one time period which applies to both files. During this time period, a transition is made from the 1st to the 2nd sound.

The 6 Modes provide a way to weight the process in favour of the salient features of the soundfiles, and which one you choose to predominate. The process also allows a further degree of weighting of the amount of the 2nd sound's frequency or amplitude interpolated at the *start* and *end* of the specified time.

Musical Applications

The application here is to move easily between two different sounds, each of which has a steady spectrum. The stability within each sound supports this kind of movement from one sound to the other. Perhaps it can be likened to the kind of smooth aural displacement that takes place when the harmony moves between triads with two notes in common (such as between G-Major and E-minor, E-minor and A-minor, etc.).

The MORPH BRIDGE process is designed to handle movement between sounds which are more (but not too) dissimilar. It takes a great deal of delicate handling to achieve these types of smooth movement between sounds: there is a need to trick the ear at the point of transition analagous to what needs to happen with any harmonic modulation. But in this case, the psycho-acoustics of the process are complex and still very little understood.

The various amplitude following options, for example used with sounds with large variations of amplitude, can be used to combine sounds in remarkable ways.

ALSO SEE: **MORPH MORPH**, where there is a somewhat more extended discussion of sound transitions.

End of MORPH BRIDGE

MORPH GLIDE – interpolate, linearly, between 2 single analysis windows

Usage

morph glide *infile1 infile2 outfile duration*

Parameters

infile1 and *infile2* – single-window analysis files which have been extracted with **SPEC GRAB**
outfile – output analysis file
duration – duration of output sound required (secs.)

Understanding the MORPH GLIDE Process

We can think of the two grabbed windows as tables of spectral data, a little snapshot cropped from the main source file. The contents will be the frequency and amplitude information for each of the e.g. 1024 channels (whichever value was entered for the number of channels). There will be different spectral data in each of these window-tables.

The data in *infile1* becomes the start value, channel by channel, and the data in *infile2* becomes the end value, channel by channel. GLIDE creates a linear interpolation between the start and end values, channel by channel, over the *duration* specified.

Note what a linear interpolation is. Think of a straight line drawn between the start and end values. The number of values and the difference between those values will depend on the *duration* specified. So we can invoke our visual imagination to create a picture for these two example situations:

1. Widely differing values over a short duration will result in dramatically rising or falling lines.
2. Scarcely differing values over a long duration will result in very subtle, gradual changes.

Because the interpolation is linear that is, proceding in equal steps the resulting sound is likely to comprise more or less gradual transitions, which is why GLIDE has been placed in the MORPH category.

Musical Applications

Whether rapid and possibly somewhat dramatic, or slow and ever so gradual, MORPH GLIDE provides a way to move between two sounds in a smooth way. But note that the transition is never likely to be as inaudibly smooth as a full spectral interpolation over time as with MORPH MORPH itself. With MORPH GLIDE we should hear and feel the sound being redirected from one state to another. The extremes of the effects possible can therefore be described, perhaps, as a forcible turning on the one hand, and a hardly noticeable translation on the other.

When two timbrally very different windows are grabbed, GLIDE can produce transitions with wonderful timbral modulations.

End of MORPH GLIDE

MORPH MORPH – morph one spectrum into another

Usage

morph morph mode *infile infile2 outfile as ae fs fe expa expf* [**-s***stagger*]

Modes

1 interpolate linearly (*exp* = 1),
or over a curve of increasing slope: slow to faster (*exp* > 1)
or decreasing slope: fast to slower (*exp* < 1)
2 interpolate over a cosinusoidal spline

Parameters

infile first of two analysis files to morph
infile2 second of two analysis files to morph
outfile output analysis file, result of the morph
as time in seconds when the amplitude interpolation begins
ae time in seconds when the amplitude interpolation ends
fs time in seconds when the frequency interpolation begins
fe time in seconds when the frequency interpolation ends
The above times refer to time position in the *infile*, i.e., the first of the two files.
expa exponent of the amplitude interpolation
expf exponent of the frequency interpolation
The *exponent* controls the speed at which the amplitude /frequency interpolation affects the *second* file, i.e., the rate at which the morph is transfering to the 2nd file
< 1 = it comes in quickly and then the rate of change slows down
1 = it comes in with a linear, steady movement
> 1 = it comes in slowly and then the rate of change speeds up
-s*stagger* time-delay of entry of 2nd file (default is 0.0)

- MORPH MORPH works only with Mono files. (For stereo/multi-channel operation, channelise the sound into mono files and process these separately.)
- If *infile* is **longer** than *infile2*, the *outfile* will terminate when *infile2* does (*infile* will be truncated).
- If *infile* is **shorter** than *infile2*, the *outfile* will continue until the end of *infile2*.
- If *stagger* is employed and *infile* is **longer** than *infile2*, the *end* time of the morph can be *stagger* plus *infile2*'s duration, but no more.
- If *stagger* is employed, the *start* time of the morph interpolation cannot be placed before the entry of *infile2*: i.e., the *start* time value cannot be less than the *stagger* value.

Understanding the MORPH MORPH Process

MORPH MORPH is a program which can be used to create smooth aural transitions between different (but fairly similar) sounds. It does this by interpolating between the values in two analysis files generated by PV to analyze the source soundfiles and re-synthesize the *outfile* of MORPH MORPH. The resulting sound will be a more or less successful interpolation between the two original source sounds.

It is important to understand that the interpolation takes place between the values in the two files *while they continue to evolve in time*. MORPH interpolates each successive window in the first file with its corresponding (simultaneous) window in the second file, so that the two sounds are continuing to unfold their original morphology during the process of interpolation.

The start and end times for the amplitude and the frequency interpolation can be set separately, and this often needs to be the case to achieve a good morph. Thinking through a morph might involve the following steps:

1. The morph could start at the beginning of each file, or one could listen for a section somewhere in the middle of the files where the spectra are fairly stable. This might be a good place to locate the morph, but ultimately one must try something and listen to the results.

It is possible that these sections do not occur at the same point in each file. Suppose you would like to place the morph between times 2 and 5 in *infile* (i.e., somewhere in the middle), and between 0 and 3 in *infile2* (i.e., at the beginning). To achieve this,

- Set the interpolation *start* and *end* times to 2 sec. and 5 sec. respectively – **NB: these times relate to the 1st input file**.
- Offset the 2nd input file by setting *stagger* to 2 sec.
- Thus the morph starts at time 2 sec. and ends at time 5 sec. in *infile* and starts at time 0 sec. and ends at time 3 sec. in *infile2*.
- You can only offset *infile2* and it can start any time before and up to (but not after) the *start* time of the morph interpolation.
- If *stagger* sets *infile2* to begin before the morph interpolation *start* time, the morph in *infile2* will begin at *start* - *stagger* sec. For example, when *start* = 1.0 and *stagger* = 0.5, the morphing of *infile2* will begin at 1.0 - 0.5 = 0.5 sec. into *infile2*, and the first 0.5 seconds of *infile2* disappears. [I could be wrong about this - AE.]
- If you set *stagger* to 2 sec. and enter an interpolation *start* time of 1 sec., it will be rejected, because it comes before the start of the 2nd file.

It is therefore possible to lay the two files against each other with considerable variety and precision:

- **no *stagger*** – the beginning of *infile* lines up with with the beginning of *infile2* **the morphing begins at the same time point in both files**

- ***stagger* = *start*** – the beginning of *infile2* lines up with the *start* of the morph interpolation in *infile*
  the morphing begins at time *start* in *infile* and at the beginning of *infile2*

- ***infile2* is positioned before *start*** – the beginning of *infile2* comes before the *start* of the morph interpolation in *infile*
  the morphing begins at time *start* in *infile* and at time *start* - *stagger* in *infile2*

  Summary: Thus the *stagger* parameter can be used to offset the file in which the section to morph comes **earlier** than the section to morph in the other file – the value of *stagger* is the amount of time in seconds that it is earlier. Make this file the second input file.

  In this way the *stagger* amount is added at the beginning of the second file (before it starts), such that this file comes into play that much later.

2. Then one considers the amplitude profile of the morph sections. It might be best to match this in time with the frequency components, or start the amplitude change a little before or a little later than the change which occurs in the frequency components.

3. *expa* and *expf* can then be used to adjust the changeover between the frequency and amplitude components of the two files. Sometimes both should move in the same way, and sometimes one wants the amplitude change to occur quicker than the frequency change, and *v.vs.* or in the reverse direction: frequency components might move slowly from the first file to the second, while the amplitude components could move more quickly. This is harder to think through than it seems, because there are in fact 9 different combinations, and of course the range of values > 1 and < 1. The following chart might provide a point of reference:

Shaping Morph Transition Characteristics with the *exponent* Parameter

Example	Param	*exponent*	Change into 2nd sound
1	amp	*expa* = 1	evenly
	frq	*expf* = 1	evenly
2	amp	*expa* > 1	slow to faster
	frq	*expf* > 1	slow to faster
3	amp	*expa* < 1	fast to slower
	frq	*expf* < 1	fast to slower
4	amp	*expa* > 1	slow to faster
	frq	*expf* < 1	fast to slower
5	amp	*expa* < 1	fast to slower
	frq	*expf* > 1	slow to faster
6	amp	*expa* = 1	evenly
	frq	*expf* > 1	slow to faster
7	amp	*expa* = 1	evenly
	frq	*expf* < 1	fast to slower
8	amp	*expa* > 1	slow to faster
	frq	*expf* = 1	evenly
9	amp	*expa* < 1	fast to slower
	frq	*expf* = 1	evenly

4. Other considerations:

- What if the two files are at different pitch levels? Try cutting out the morphing section of one of the files, creating a time-varying transposition (probably in the spectral domain) to the level of the other file and replacing it before doing the morph.

- What if one file is pulsed and the other isn't? Try extracting the amplitude envelope of the pulsed one and imposing it on the other file before doing the morph. (See **ENVEL EXTRACT** and **ENVEL IMPOSE**).

- What if one of the files has such a variable spectrum that it interferes with the smoothness of the morph? Try a time varying spectral BLUR starting at the point where the morph is to begin.

- What if the two sounds have distinctly different formant components? Try extracting the formants from one of the sounds and imposing them on the other.

Musical Applications

We are now familiar with the practice of gradual shape transformations between visual images. The process has a wonderful surprise value as the first image seems to alter before our very eyes, becoming another quite different image, such as a face becoming that of another person.

You may also possibly have observed that just at the moment of transition, the first image begins to melt or distort in some way. I say 'possibly' because it tends to happen very quickly, and because our eyes/brain tend to construct a familiar image, it's rather difficult to notice what happens inbetween. However, it is very instructive to do so. If it didn't change in some way before the second image made its presence felt, there would in fact not be a transition, just an abrupt change. This is the principle of 'modulation' in action.

– which is precisely where we need to turn our attention when seeking to carry out an audio-morph. Modulation in music has been used up till now to move from one tonal region to another, and you will recall how it is common practice to begin by dissolving the identity of the existing tonal region, such as by using tonally ambigous diminished 7th harmonies.

The idea of creating surprise and astonishment by causing one sound to become another is in itself easy to grasp. What's not so easy is making this smooth to the point of being almost imperceptible. Herein lies the art of the morph, and once again the concept of modulation is relevant. ('Morph', by the way, is a Greek word meaning 'form'.) This program gives a taste of the morphing process. The CDP spectral domain programs provide many other processes that can be used to 'pre-process' sounds to achieve a more psycho-acoustically effective transition.

Here, the best results will be achieved by using sounds which are already quite similar.

Historical note

MORPH is derived from VOCINTE, which was originally written at IRCAM during the composition of VOX-5. I would like to thank the many people there who offered me help and advice. I would also like to thank Andrew Bentley who modified the first versions of the program to work with the (Atari-based) CDP soundfile system and to operate more efficiently. (Trevor Wishart)

End of MORPH MORPH

NEWMORPH – Morph between dissimilar spectra

Usage

newmorph newmorph 1-6 *inanalfile1 inanalfile2 outanalfile stagger startmorph endmorph exponent peaks* [**-e**] [**-n**] [**-f**]
newmorph newmorph 7 *inanalfile1 inanalfile2 outanalfile peaks outcnt* [**-e**] [**-n**] [**-f**]

Example command line to create a timbral morph:

```
newmorph newmorph 1 in1.ana in2.ana out.ana 0 0 2 5 5
```

Modes

1 Interpolate linearly (*exponent* = 1) between the average peak channels or over a curve of increasing (*exponent* > 1) or decreasing (exp <1) slope, simultaneously moving spectral peaks, and interpolating all remaining channels.
2 Interpolate cosinusoidally (*exponent* = 1) between the average peak channels or over a warped cosinusoidal spline (*exponent* not equal to 1), simultaneously moving spectral peaks, and interpolating all remaining channels.
3 As mode 1, using channel-by-channel calculation of peaks.
4 As mode 2, using channel-by-channel calculation of peaks.
6 Sound 1 is (gradually) tuned to the (averaged) harmonic field of sound 2 linearly
6 Sound 1 is (gradually) tuned to the (averaged) harmonic field of sound 2 cosinusoidally
7 Sound 1 is morphed towards sound2 in *outcnt* steps, each step a new output file.

Parameters

inanalfile1, inanalfile2 – input analysis files made with PVOC
outanalfile – output analysis file
stagger – time before the entry of analysis file2
startmorph – time in the 1st file at which the morph starts
endmorph – time in the 1st file at which the morph ends
exponent – exponent of interpolation (Range: 0.02 to 50)
peaks – number of peaks to interpolate (Range: 1 to 16)
outcnt – **Mode** 7 makes *outcnt* distinct ouput analysis files
-e – retain the loudness envelope of the first sound. In this case the *outanalfile* is terminated when the end of the first sound is reached.
-n – there is no interpolation of anything except the peaks
-f – only frequency is determined by the peaks in analysisfile2

The end of the interpolation must not be beyond the end of one of the files.

Understanding the NEWMORPH Process

NEWMORPH is similar to MORPH MORPH in that there are two input analysis files and there is a gradual movement from the first to the second, with start and end times for the morph and *exponent* and *stagger* parameters. Where it differs is that NEWMORPH's morph is focused on the peaks of the two spectra. This helps the program to handle dissimilar spectra, whereas MORPH MORPH works best with similar sounds. There are also a larger number of modes, adding channel-by-channel operation, tuning and morphing in steps.

End of NEWMORPH

NEWMORPH2 – Morph frequencies of spectral peaks

Usage

newmorph newmorph2 1 *inanalfileA outtextfile* (=*peaksfile* in Modes 2-3) *peakcnt*
newmorph newmorph2 2-3 *inanalfileB peaksfile outanalfile startmorph endmorph exponent peakcnt* [**-r***rand*]

Example command line to create a tuned morph:

```
newmorph newmorph 1 in.ana peaksfile.txt out.ana 0 2 1 5
```

Modes

1 Find frequencies of the most prominent spectral peaks of inanalfileA and output these in order of prominence, as a textfile list: a *peaksfile*
2 The sound of inanalfileB is (gradually) tuned to the harmonic field specified in the *peaksfile***1**. The *peaksfile* lists the goal-peak frequencies, in order of prominence.
3 As Mode 2, but the interpolation is cosinusoidal over time instead of linear.

Parameters

Mode **1**:
inanalfileA – input analysis file in which to find the frequencies of spectral peaks
outtextfile – output textfile (*peaksfile*), listing the frequencies of the most prominent peaks, in order of prominence
peakcnt – number of most-prominent peaks to find (Range: 1 to 16)

Modes **2** and **3**:
inanalfileB – input analysis file to which to apply the data in the *peaksfile*
outanalfile – output analysis file
peaksfile – input textfile (from Modes **1**), listing the frequencies of the most prominent peaks in order of prominence; **OR** a textfile containing an arbitrary list of frequencies, not necessarily those extracted from a previous file in Mode **1**
startmorph – time at which the morph starts
endmorph – time at which the morph ends
exponent – exponent of interpolation (Range: 0.02 to 50). See MORPH MORPH for a discussion of this parameter.
-rrand – randomisation of the goal peak frequency (Range: 0 to 1)
peakcnt – number of most-prominent peaks to retune (Range: 1 to 16)

Understanding the NEWMORPH2 Process

NEWMORPH2 aims to morph the most prominent frequencies of one sound towards those of another. The first pass (Mode 1) extracts the goal frequencies as a text list which becomes the *peaksfile* for Mode 2 or 3. In practice, however, the *peaksfile* can be any list of frequencies (up to 16), listed in order of prominence. NEWMORPH2 is similar to NEWMORPH's tuned modes (6 and 7), but appears to be more focussed on the spectral peaks found in the goal sound.

End of NEWMORPH2