VCI4DMI MANUAL (draft 0.1 version 15/7/2014)

Voice-Controlled Interface (VCI) for Digital Musical Instruments (DMI)

The Voice-Controlled Interface for Digital Musical Instruments (VCI4DMI) is a collection of MAX patches, MAX externals and MATLAB functions implementing ad-hoc mappings to control an arbitrary number of real-valued instrument parameters by variation of the voice timbre.

The VCI4DMI can be obtained at http://stefanofasciani.com/vci4dmi.html

Inquiries: stefanofasciani@stefanofasciani.com

The VCI4DMI is implemented in MATLAB, Max/MSP, and Ableton Live (for instrument hosting). These software need to be installed in order to run the VCI4DMI.

The VCI4DMI has been developed and tested on the following platform:
OSX 10.7.5
MAX 6.1.5 (32-bit)
MATLAB 7.12.0 (R2011a)
ABLETON LIVE 9.1 (32-bit)
FTM 2.6.0 beta

The implementation should be compatible with higher version of the software mentioned above. For 64-bit versions of MAX and Ableton Live the custom FTM externals binaries has to be re-built. The VCI4DMI has never been ported or tested on Windows. The Max patches and MATLAB code should be compatible but the custom FTM externals binaries must be built for the specific operative system.

OSX users without a Live, Max or MATLAB license can still test and try the VCI4DMI:
- install a demo of Ableton Live;
- install the 30 days trial of Max which comes with the Max Runtime application which allows running but not editing Max patches (also after the trial expiration);
- ask via email for a compiled version of the MATLAB functions into executable that can be run from Terminal (this may not work on all systems);

INSTALLATION

1) Download and install the following third party MAX & MATLAB libraries/toolboxes:

for MAX:

- FTM: http://ftm.ircam.fr/index.php/Main_Page install or copy the externals and abstractions in a folder included in the MAX path.
- analyzer~: http://web.media.mit.edu/~tristan/maxmsp.html copy the external in a folder included in the MAX path.
- netsend~: http://www.remu.fr/sound-delta/netsend~/ copy the externals (netsend~.mxo and netreceive~.mxo) in a folder included in the MAX path (version 1.0b).
- OSC-route: http://cnmat.berkeley.edu/patch/4029 copy the external in a folder included in the MAX path.

for MATLAB:

- drtoolbox: http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html include the folder and subfolders in the MATLAB path.

- distmesh: http://persson.berkeley.edu/distmesh/ include the folder and subfolders in the MATLAB path.
- rastamat: http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/ include the folder and subfolders in the MATLAB path.
- oscmex: http://sourceforge.jp/projects/sfnet_oscmex/ include the folder and subfolders in the MATLAB path (it requires liblo installed http://liblo.sourceforge.net)
- Neural Network Toolbox: http://www.mathworks.com/products/neural-network/ the toolbox properly installed in MATLAB is necessary to run the VCI4DMI DMI mapping learning.
- Parallel Computing Toolbox: http://www.mathworks.com/products/parallel-computing/?s_cid=sol_compbio_sub2_relprod4_parallel_computing_toolbox the toolbox properly installed in MATLAB is necessary to run the parallel version (non plot) of the MATLAB part of the runtime VCI4DMI.

2) Install VCI4DMI MAX & MATLAB library

for MAX:

- copy the abstractions in MaxLib/Abstractions into a folder included in the MAX path.
- copy the FTM externals in MaxLib/FTMexernals into your MAX FTM externals folder (which is likely to be /Applications/Max 6.1/Cycling '74/ftm-and-co/externals if you used the FTM installer). For running on 64-bit versions of Max or on Windows the externals but be re-built and the related source code are included in MaxLib/FTMexternals-sources (the provided one are built for 32-bit Max with FTMEXT-SDK-2-6-MacOSX and MaxSDK-6.1.1).

for MATLAB:

- copy the .m files in MATLABLib folder into a folder included in the MATLAB path.

3) Import the MaxForLive devices.

Import the four MaxForLive devices contained in the sub folders of (two Max Audio Effects, two Max Midi Effects) contained in the subfolders of MaxForLive/Presets in your Ableton Live 9 user library. You can either copy the files manually into the User Library related sub folders, or open the .amxd in Live and click on the save icon on the top right of the MaxForLive device.

4) MAX & MATLAB main patches/function

for MAX (in the folder Max):

- VCI4DMI-DMIanalysis.maxpat online part of the instrument analysis, interact with MaxForLive Devices.
- VCI4DMI-Runtime.maxpat real-time voice analysis for the runtime interface and partial sub optimal vocal GC training.

for MATLAB (in the folder MATLAB):

- VoiceFeatBlindSearch.m offline learning for the optimal features computation settings and noisy features rejection.
- SOGgcLearn.m offline learning stage for the generate the vocal gestural controller using the self-organizing gesture technique, generates mapping for the runtime part.
- SOGgcLearnMLDA.m variation of the offline learning stage for the generate the vocal gestural controller using the self-organizing gesture technique with the multiclass LDA for dimensionality reduction instead of Isomap, generates mapping for the runtime part.
- DMImappingFreq.m offline learning of the DMI mapping for sound generators and frequency domain processors, generates mapping for the runtime part.
- DMImappingTime.m offline learning of the DMI mapping for time domain processors, generates mapping for the runtime part.
- DMImappingFreqExtFeat.m offline learning utility to compute the DMI mapping for sound generators and frequency domain processors using timbre features computed with an external library.

- runtimeVCI4DMI.m runtime part of the vocal interface, non parallel version, lower max throughput voice to DMI parameters, optional visual feedback of vocal and instrument spaces
- runtimeVCI4DMIhiperf.m runtime part of the vocal interface, parallel version requires MATLAB parallel computation toolbox, higher max throughput voice to DMI parameters, no visual feedback

BASIC USE and EXAMPLES

For the current version of the VCI4DMI both Max and Ableton Live sampling rate must be set to 48KHz.

The VCI4DMI requires:
- VOCAL GESTURAL CONTROLLER TRAINING: a training procedure for learning how to extract a number of intermediate continuous control signals (in this version limited to 2 or 3) from specific performer vocal gestures. Vocal gestures are meant to be continuous variation of the voice timbre (pitch and loudness variation are not relevant) e.g. gliding between different vowels. These signals are then used to control an arbitrary set of real-valued instrument parameters.
- DMI MAPPING TRAINING: an analysis and training procedure to generate a representation of how the perception of sound of a specific instrument (a sound generator/synthesizer or processor/effect) changes with varying an arbitrary number (up to 8 in this version) of real-valued instrument parameters (target of the vocal control). This is used to implement a few to many mapping implemented in the perceptual sonic space of the instrument. This implies that the control signals computed from the voice are coordinates for browsing/navigating in the sonic space of the instrument, from which the system retrieve the set of instrument parameter associated with the target sound.
- Pairing 2 outcomes of the previous training procedures and run in real time the vocal interface. The outcome of the previous steps are independent and can be paired or exchanged arbitrarily. The instrument training procedure generates a mapping for 2D and 3D spaces, so it is compatible with any trained vocal gestural controller.
- The training and mapping procedure are unsupervised. This minimizes the user effort in generating the training data, but require the user to get familiar with the outcome of the training procedures. The visual feedback provided by the system ease the understanding of the unsupervisedly generated mapping. The Max GUI provides plenty of options to tune the generated vocal gestural controller and mapping according to the user preferences. Note that the system does not perform the voice-to-instrument-sound mapping based on timbre similarity, but the mapping can be further tuned in runtime to this direction inverting the output ranges of the vocal gestural controller output signals.

For mode details about theory behind the VCI4DMI refer to Chapters 3-5 of Fasciani, S. 2014. "Voice-Controlled Interface for Digital Musical Instruments". PhD Thesis, National University of Singapore. Chapter 6 contains details about the prototype implementation, operational modes, Max patches GUI options, non included in this manual. The thesis will be published online in the second half of October 2014. Excerpt of chapters 3-5 can be requested via email, or refer to Fasciani, S. and Wyse, L. 2013. One at a Time by Voice: Performing with the Voice-Controlled Interface for Digital Musical Instruments. In Proceedings of the NTU/ADM Symposium on Sound and Interactivity 2013, Singapore, extended for the eContact! 16.2, 2014.

STEP 1 – Offline Vocal Gestural Controller training procedure

Create a folder that will be used to store voice training data and training procedure outcome. For this guide we suppose the folder name is “MyVocalGC” and the absolute path to the folder is “/Users/stefano/VCI4DMI/MyVocalGC”.

Record a set of vocal postures (sustained invariant voice timbres) into different wave files and name them “post1.wav”, “post2.wav”, etc. There is no theoretical maximum or minimum number of files or temporal duration of the postures files. However it is advisable to provide at least 4 postures files each lasting at least 5 seconds each. The vocal posture timbres should represents steady voice timbres between which the vocal gestures glide. Put the files into the “MyVocalGC” folder.

Record different instances of vocal gestures (variation of voice timbre) and marge these into a single file named “gest.wav”. The vocal gestures should include only variation of the voice timbre, avoiding the presence invariant timbres. The variations may include gliding between different invariant timbres (such as gliding between different vowel sounds). The user should consider the gestures will be used to browse or navigate into a 2D or 3D space, thus in the “gest.wav” file an equal amount of time should be dedicated to examples of specific timbre variation, ideally uniformly covering a voice timbre space. The rate or speed of the variation of the voice timbre affects the outcome of the training of the vocal gestural controller, a mixture of slow and fast variations is recommended for novice users.

Vocal postures and gesture files must be mono and 48KHz. It is recommended to edit audio files to avoid presence of silence or pauses at the beginning, end, or within the files. In recording the vocal postures and gestures it is recommended to use the same recording devices (microphone and soundcard) that will be used for performing. Recording to wave files can be performed also using the VCI4DMI-Runtime.maxpat section showed in Figure 1 (open file and record at the bottom), in which is possible to optionally select gain, compressor and filter to the input signal (performing settings should be the same).

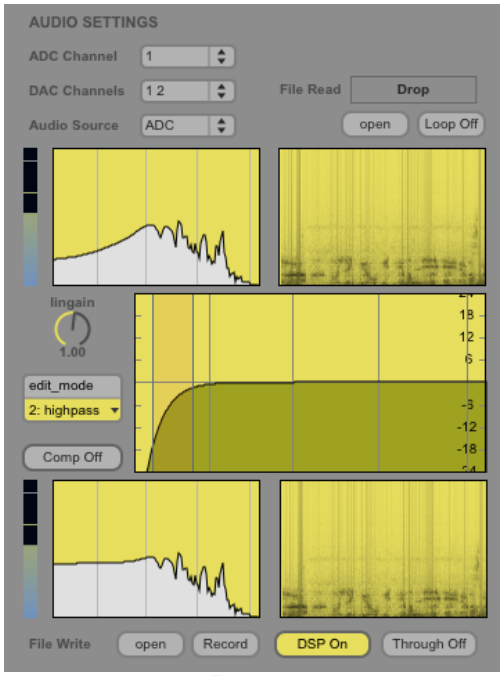

Figure 1

In MATLAB run the function `VoiceFeatBlindSearch(path,numpost,full)` where:
- path is the absolute path to the “MyVocalGC” folder
- numpost if the numper of post#.wav files in the folder “MyVocalGC”
- full is a flag that determines the number of search combinations, if equal to 1 the algorithm search on 400+ combinations, while if 0 it search is performed on a drastic lower number of combination, resulting in faster execution but sub optimal solution.
(as an alternative a sub-sub optimal solution with user defined feature computation setting can be computed directly in the VCI4DMI-Runtime.maxpat INSTRUCTION TBC)

In MATLAB run the function `SOGgcLearn(path,varargin)` where:
- path is the absolute path to the “MyVocalGC” folder (the function use as input the files generated from the VoiceFeatBlindSearch function).

- the second argument can force the dimensionality of the gestural controller output to either 2 or 3, if not defined the dimensionality is derived from the intrinsic dimensionality found in the vocal gestures.
- the third argument represent the pre filtering option. If not specified or if equal to -1 pre filtering is not performed, if 0 the pre filtering is performed on the most likely number of cluster detected in the vocal gesture data, other positive integers determine manually the numbers of clusters in the vocal gesture data. The number of cluster should be equal to the number of vocal postures included in the vocal gesture.
- for the other arguments refer to the source code of the SOGgcLearn.m file.
(as an alternative it is possible to use the function `SOGgcLearnMLDA(path,varargin)` *in which the Isomap dimensionality reduction is replaced by a multiclass LDA, and 4 or 8 vocal postures will be associated with the vertices of the gestural controller output space, respectively in 2D or 3D; in this case the second argument will be the number of vocal posture to consider as gestural extrema, which has to be a power of two; if equal to 4 the postures files 1-4 will be associated to the extrema, while if equal to 8 the postures files 1-8 will be associated with the extrema; the user can not specify the individual associations posture-extrema-vertices; the method does not guarantee the association if the voice timbres in the postures files and gesture file are inconsistent).*

Examples
In the folder "VCI4DMI/Examples/VocalGC'" there are examples of vocal gestural controllers derived from a set of 5 vocal postures (steady/invariant vowel sounds) and vocal gesture (gliding between vowels).

These were generated using the following commands in MATLAB:

```
VoiceFeatBlindSearch('/Users/stefano/VCI4DMI/Examples/VocalGC',5,1);
SOGgcLearn('/Users/stefano/VCI4DMI/VCI4DMI/Examples/VocalGC',3,0);
SOGgcLearn('/Users/stefano/VCI4DMI/VCI4DMI/Examples/VocalGC',2,0);
SOGgcLearnMLDA('/Users/stefano/VCI4DMI/VCI4DMI/Examples/VocalGC',4,0);
```

The three gestural controllers are in a data structure saved in the files mapV2.mat (2D vocal gestural controller), mapV3.mat (3D vocal gestural controller), mapVwp2.mat (2D vocal gestural controller computed with the multiclass LDA variance associating the postures 1-4 to the vertices of the gestural controller output space).

All other files in the "VCI4DMI/Examples/VocalGC" folder are generated by the execution of the above-mentioned functions.

If you run out of RAM during the computation of the 3D vocal gestural controller, comment the plot commands in the SOGgcLearn function and sub functions. These speeds up the computation as well, or reduce the number of training iterations.

STEP 2 – Online DMI analysis

Load your instrument in Ableton Live and wrap it with the Front-End (VCI4DMI-GeneratorsFrontEnd.amxd or VCI4DMI-ProcessorsFrontEnd.amxd) and Back-End (VCI4DMI-BackEnd.amxd) devices as in Figure 2. Enable MIDI and Audio Through if you want the Live track messages to go through the devices (with Audio Through disabled you won't hear the instrument sound and with MIDI through disabled you wont be able to drive the instrument with MIDI messages).

Figure 2

In the Front-End select the target instrument from the left drop down menu (press list first) and then select up to 8 target control parameters on the instrument from the right drop down menus. Upon selecting the parameters these become grey and unresponsive in the DMI, but these can still be controlled from the fader+number GUI elements of the right of the parameters drop down menus. The on-off toggle next to the parameters is set automatically via OSC from the VCI4DMI-DMIanalysis.maxpat, and does not affect the manual control of parameters within live. Set the OSC receive port to the same value as the OSC send port in the VCI4DMI-DMIanalysis.maxpat.

In the Back-End select the IP address of the machine running the VCI4DMI-DMIanalysis.maxpat (127.0.0.1 if on the same machine) and specify the netsend communication port (8008 by default in the VCI4DMI-DMIanalysis.maxpat). When the VCI4DMI-DMIanalysis.maxpat is loaded press connect to establish the connection between the Back-End and the VCI4DMI-DMIanalysis.maxpat. Ensure that the box at the bottom is in “Connected”.

In the VCI4DMI-DMIanalysis.maxpat:
-set the OSC IP address and port to communicate with the Front-End maxpat (127.0.0.1 if on the same machine) as in Figure 3.

Figure 3

- enable the active target DMI parameters by enabling the on/off button for each parameter (these are in the same left-to-right order to the top-to-bottom order in the Front-End). For each parameter set minimum, maximum, and step resolution value used for the analysis, as in Figure 4. Note that the ranges are normalized to 0-1 and automatically scaled to the right parameter range in the MaxForLive Front-End. Note that the active parameters must be consecutive. Push “Update Enabled” to reflect the changes in the Front-End (not mandatory, it happens automatically at analysis startup) and push “Update # Combs” to visualize on left the total number of combinations to analyze (not mandatory, it happens automatically at analysis startup). Consider that the higher the combinations the longer the analysis time takes (which depends also by the analysis settings). The analysis time is estimated on the right side of the VCI4DMI-DMIanalysis.maxpat. The maximum values must be bigger than the minimum otherwise the analysis will not start.

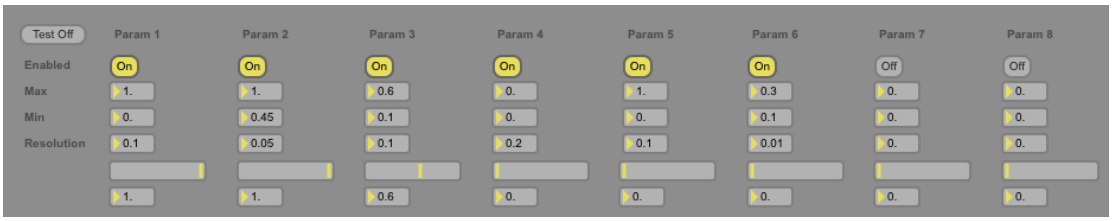

Figure 4

- set the analysis options in the bottom part as in Figure 5. Note that the Analysis Mode drop down menu contains 4 choices: “Gen Sustain” is for sound generator with sustained or variable timbre; ”Gen Retrig” is for sound generators with decaying timbre; “Proc Noise” is for sound processor analysis in frequency domain (using same features as generators) but using one of the available noises as signal source; “Proc Impulse” is for sound processor analysis in time domain (impulse response based measurements).

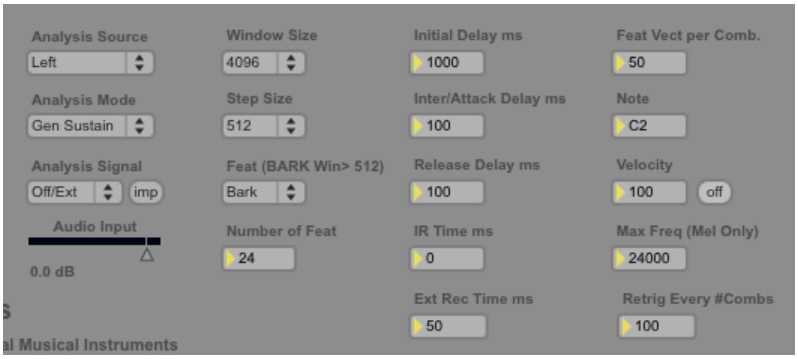

Figure 5

- for all analysis mode except the “Proc Impulse”, the frequency domain based features (timbre descriptors) can be individually disabled/enabled (masked/unmasked) by clicking on the visual representation (the related part of the display turn grey). Clicking on the top part of the display the mask is turned on and vice versa. In Figure 6 an example of partially disabled bark and spectral moments. Note that for the analysis in the time domain this feature is not available. Mind that for a correct visualization the Max and Min values on the right side of the Mel/Bark plot must be set manually.

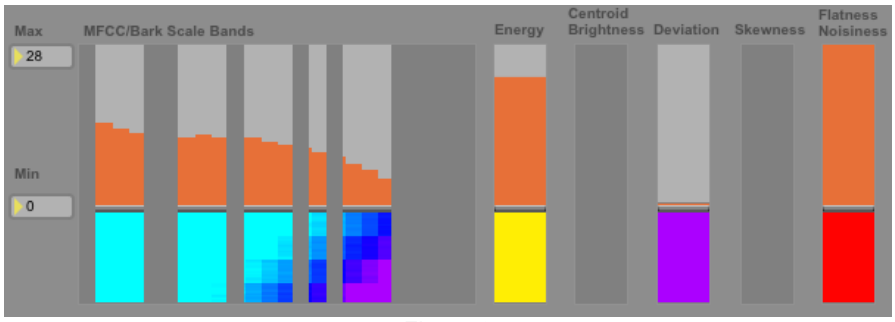

Figure 6

- on the top of the patch GUI it is possible to enable the test mode (which need to be disabled to run the analysis). In test mode the descriptor computation and visualization is enabled and the DMI parameters can be controlled with the sliders on the top of the patch GUI.

- To start the analysis press the button on top left (which can be also used to abort/interrupt the analysis). Before to start ensure that the Front-End OSC connection and the Back-End audio connection are established, and the DMI audio output is flowing into the VCI4DMI-

DMIanalysis.maxpat audio input (input meter on the bottom left of the patch). All the settings can be saved stored and recalled into a preset.

- When the analysis ends create a folder “MyDMIMapping” and export parameters matrix and features matrix into two text files using the two “Export” buttons as in Figure 7. These must be named “params.txt” and “feat.txt” respectively. The size of feat.txt can be large, if so is possible to zip the file into feat.txt.zip and delete the feat.txt file (the following MATLAB function provide to temporary extract the compressed file).

Figure 7

Examples
In the folder “VCI4DMI/Examples” there are four subfolders (DMIgeneratorSustained, DMIgeneratorDecaying, DMIprocessorFreq, DMIprocessorTime) each containing an Ableton Live project. Each project contains a different DMI with Front-End, and Back-End configured for the analysis. The four DMIs can be directly analyzed with the VCI4DMI-DMIanalysis.maxpat recalling one of the last four presets (name of the project displayed below the preset after recall). Each folder contains also the files “feat.txt” (zipped) and “params.txt” outcome of the analysis stage. For these examples the number of variable parameters is limited to 2 or 4 and the parameters step resolution is large in order to minimize the analysis time (5-15 minutes).

STEP 3 – Offline DMI mapping training procedure

The DMI mapping generation is performed in a MATLAB function that requires as argument the absolute path of the folder containing the files “feat.txt” and “params.txt”, some of the online analysis options, the offline analysis mode and the parameters kernel flag.

For sound generators or sound processors analyzed in the frequency domain (noise as input signal) use the function
`DMImappingFreq(path,vectperstate,winsize,hopsize,kernel,mode)` where:
- path is the absolute path to the folder containing “feat.txt” and “params.txt”
- vectperstate is the number of timbre descriptor vectors per each state (or combination) of DMI input parameters set in the in the VCI4DMI-DMIanalysis.maxpat online analysis options
- winsize is the analysis windows size of set in the in the VCI4DMI-DMIanalysis.maxpat online analysis options
- hopsize is the analysis windows step of set in the in the VCI4DMI-DMIanalysis.maxpat online analysis options
- kernel is a flag that is non zero add to the timbre descriptor feature a scalar derived from a kernel function applied to the DMI parameter combination. This helps to discriminate in the sonic representation between identical or similar sounds generated with very different parameter combination set.
- mode: if equal to 0 the vectperstate are averaged, and this mode is dedicated to sound generators/processors with steady sustained timbre; if equal to 1 the vectperstate are used to determine maximum, minimum and period (if any), and this mode is dedicated to sound generators/processors with variable sustained timbre; if equal to 2 the whole vectperstate

sequence (representing the timbre descriptors over the attack-decay envelope) are used for the DMI mapping generation, and this mode is dedicated to sound generators with decaying timbre.

For sound processors analyzed in the time domain (impulse as input signal) use the function `DMImappingTime(path,kernel,mode)` where:
- path is the absolute path to the folder containing “feat.txt” and “params.txt”
- kernel is a flag that is non zero add to the timbre descriptor feature a scalar derived from a kernel function applied to the DMI parameter combination. This helps to discriminate in the sonic representation between identical or similar sounds generated with very different parameter combination set.
- mode: if equal to 0 the whole impulse response (internally down-sampled typically to 8KHz) is used as descriptor vector, and should be used with DMI such as delay where the impulse response includes only few replicas of the original impulse; if equal to 1 a set of descriptors are computed from the impulse response (such as energy, T60, slope,…) and this should be used with DMI such as reverberators that generate more dense impulse response.

Both functions stores the DMI mapping in a data structure saved in a .mat file in the same folder specified by the path argument.

Examples
In the four sub-floders of VCI4DMI/Examples” (DMIgeneratorSustained, DMIgeneratorDecaying, DMIprocessorFreq, DMIprocessorTime) the DMI mapping .mat files was generated with the following MATLAB commands:

```
DMImappingFreq('/Users/stefano/VCI4DMI/Examples/DMIgeneratorSustained',50,4096,512,0,0);
DMImappingFreq('/Users/stefano/VCI4DMI/Examples/DMIgeneratorDecaying',40,4096,1024,0,2);
DMImappingFreq('/Users/Stefano/VCI4DMI/Examples/DMIprocessorFreq',16,4096,1024,0,0);
DMImappingTime('/Users/stefano/VCI4DMI/Examples/DMIprocessorTime',0,0);
```

The DMI mapping data structure can be directly used in the runtime VCI4DMI.

STEP 4 – Preparing the VCI4DMI banks of vocal GC and DMI mapping and a Live project with multiple DMI target of the vocal control

The vocal gestural controller data structures and the DMI mapping data structures must be organized is two banks that are then loaded in the MATLAB part of the runtime VCI4DMI. This allow to swap in less than 1 second the vocal gestural controller and/or the DMI mapping loaded into the interface without the need of restarting it. However if the user requires a single pair vocal gestural controller – DMI mapping, he can load a single data structure in each bank.
The bank is a MATLAB cell array that contain an arbitrary number of prepare the bank. In the previous sections we described how to generate the vocal gestural controller and DMI mapping, both stored in the respective .mat file.
To prepare a bank simply load the .mat file. This contains a data structure called “mapV” for the vocal gestural controller, and “mapD” for the DMI mapping. In the MATLAB command prompt build the cell array bank assigning to each cell a data structure. (e.g. “bankV{1}=mapV”, load another .mat file then “bankV{2}=mapV”…. And for the DMI “bankD{2}=mapD”, …).
To retrieve which vocal gestural controller or DMI mapping was associated with a specific bank entry it is possible to display the name field of each bank entry, entering the following command in the MATLAB command prompt “bankV{1}.name”, or “bankD{4}.name”
Note that the MATLAB runtime part of the VCI4DMI communicates with Max and Live using the OSC protocol. At lines 9-13 of runtimeVCI4DMI.m or runtimeVCI4DMIhiperf.m it is possible to specify the IP addresses and the OSC send and receive ports. The default values are compatibles with the examples provided and described in this manual. Note that for the

communication with Live there is a OSC base port. This means that if more than one DMI mapping is present in the bank, the instrument associated with the bankD{1} will be driven on the base port, the bankD{2} on base port +1, the bankD{2} on base port +2. Therefore the Live project containing multiple target DMI should be properly configured (right port in the Front-End). However the target OSC port can be changed in a second stage in the VCI4DMI-Runtime.maxpat.

Examples
In the folder "VCI4DMI/Examples/Runtime" we there is a bank.mat file that includes two MATLAB cell array: "bankV" which includes the three vocal gestural controllers computed in the Sep 1 Examples, and "bankD" which includes the four DMI mappings computed in the Step 3 Examples.
In the folder "VCI4DMI/Examples/Runtime" there is the Live project "DMIset", which includes all the DMIs analyzed and for which we generated the mappings with the training procedure described above. The Back-End are not necessary for performing with the system, thus are not present in the Live tracks device chain. The Front-End associated with the individual DMIs have different OSC port value, in the same order as we associated the individual DMI mapping data structure in the bankD cell array in MATLAB.

STEP 5 – Running the VCI4DMIPreparing the VCI4DMI banks of

Open the Live project containing the DMI set. Open the VCI4DMI-Runtime.maxpat and ensure that the OSC ports are the same as those specified in the file runtimeVCI4DMI.m (lines 9-13). In Matlab run the function `runtimeVCI4DMI(bankV,bankD)` where the two arguments are the banks (MATLAB cell array) of vocal gestural controller and DMI mapping. After the system is completely controlled (including termination of the MATLAB function) from the GUI of VCI4DMI-Runtime.maxpat.

In the VCI4DMI-Runtime.maxpat enable the DSP, select the RT-control mode, and Start the Analysis, as in Figure 8. The RT-control mode push buttons allows to: stop the MATLAB function runtimeVCI4DMI, enable/disable the MATLAB interactive plotting of the vocal gesture space and dmi sonic space (in Figure 9), force the store the audio settings modified configuration in the vocal gestural controller data structure in MATLAB, enable/disable the gestural controller output and dmi parameters visual feedback on the right of the VCI4DMI-Runtime.maxpat (in Figure 10).

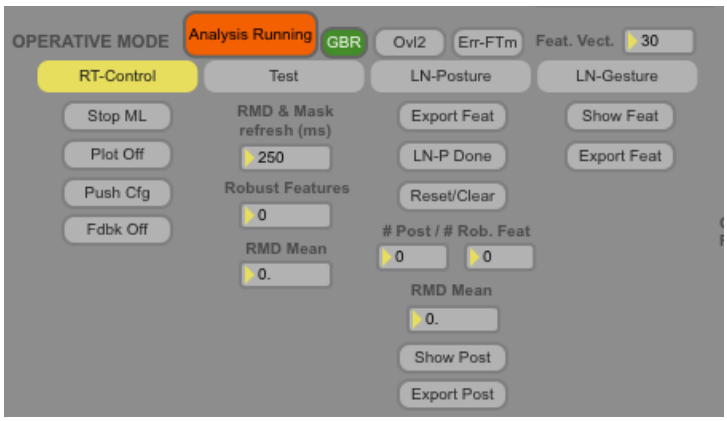

Figure 8

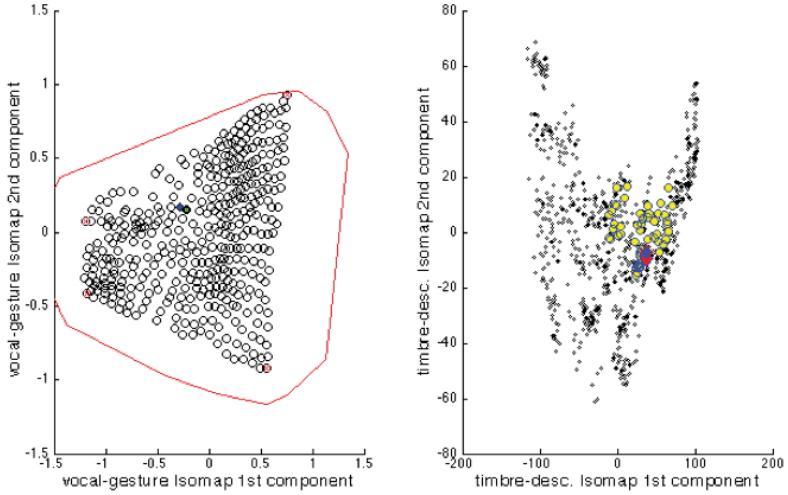

Figure 9

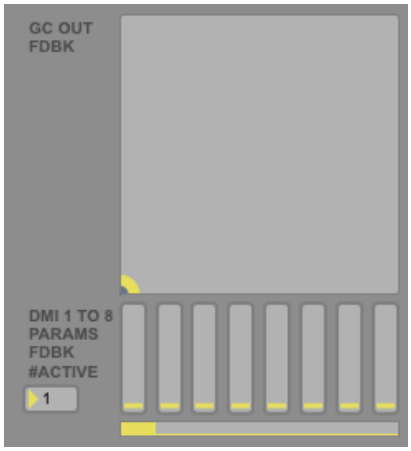

Figure 10

Select the indexes (or ID) for the vocal gestural controller (mapV) and for the DMI mapping (mapD) (indexes are related to the bankV and bankD MATLAB cell arrays) as in Figure 11 and send the indexes to MATLAB by pressing “SwitchMapID”. Pair of maps ID can be saved in preset for faster recall (recalling a preset automatically sends both ID to MATLAB), and this can be performed anytime during runtime (the interface is disabled for about 1 second to get reconfigured)

Figure 11

The system settings in Figure 12 are automatically set by MATLAB each time then when loading a new vocal gestural controller. However these can be changed manually (but should not, especially for the feature orders) and modification will be stored in the MATLAB data structure. Reducing or increasing the window step increases or reduces the rate DMI parameter vectors generation. To achieve higher rates (faster execution) use the function `runtimeVCI4DMIhiperf(bankV,bankD)` but mind that the function does not allow the spaces representation of Figure 9, but only the feedback of figure 10 (use the slower version

with plots to get familiar with the mappings and to tune them with the options discussed later, and user the high performance version only when performing).

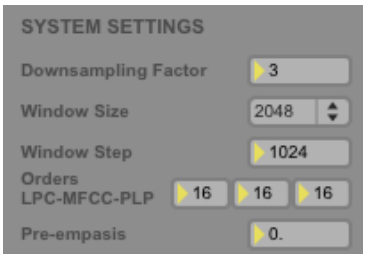

Figure 12

Faster execution or higher throughput can be achieved disabling the graphical representation of the low-level voice features with the dedicated button as in Figure 13. Removing from presentation mode other visual representation (such as spectrum and spectrogram of the incoming audio signal) increase the maximum voice-to-parameters achievable rate (lowering the window step size). In Figure 13 there's also a dial controlling the coefficient of a low pass filter applied to the low-level voice features before these are sent to MALTAB. High coefficients smooth (but it also slow down) the response of the system.

Figure 13

Vocal gestural controller operational mode settings and options for changing and adjusting the response of the gestural controller at runtime are controlled from the section of the GUI shown in Figure 14. The search can be extended to the whole output lattice or restricted to the neighbor only, with optional gravitational attraction search metric, with user defined gravitational constant. The convex hull validity check can be enabled, disabled, and optionally used to orthogonally project on the boundary vectors falling outside the hull. Two scaling factors are available to expand and contract the convex hull and the SOG lattice. The status of the interpolation by IDW can be changed as well as the power parameter. The user can change the voice to sonic space mapping by inverting the output ranges of the individual gestural controller output signal. An optional adaptive mode is available to modify the GC after every performer's voice interruption. In particular we shift the SOG lattice weights to match the new input vocal-posture spatial coordinate with the closer lattice vertex. Finally the GC can be bypassed to browse the DMI sonic space directly with the low-level voice features.

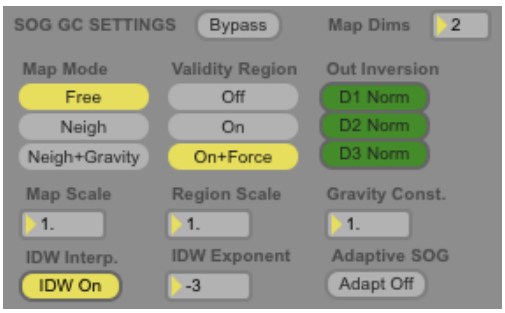

Figure 14

DMI mapping operational mode settings, in Figure 15, and options change sensitivity and response of the instrument dependent interface component. The mapping function can be set to the nonlinear ANN, direct linear scaling plus offset, or bypassed browsing directly the uniformly redistributed space. The restricted search space can be set to the entire sonic space, to the first or second neighborhood level or within a radius distance from the previous output vector od DMI parameter. The DMI parameters IDW interpolation status can be changed as well as the power parameter. Finally the DMI mapping can be bypassed linking the vocal gestural controller output directly to the to the instrument parameters. The radius can be dynamically related to the instantaneous voice variation, to improve the DMI output parameters stability over voice- postures.

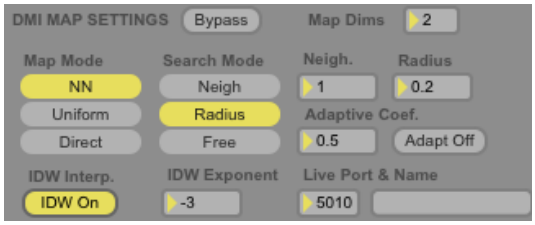

Figure 15

Two timed gates are provided to perform or inhibit the mapping based on the live input voice characteristics, shown in Figure 16. The first is based on energy level and optionally also on the detection of voiced sound. The second works on the low-level voice features flux, which is the instantaneous Euclidean distance of two consecutives voice low-level feature vectors. For both gates the user can set the threshold, monitor the real-time value, and set the minimum true condition time interval to open the gate. The interface can still react when just background noise at the input generates a voice coordinates inside the convex hull, thus the gates provides a temporary interruption of the interface mapping when voice is not present at the input, or when the voice is invariant. The timed gate is effective with burst noise such as background music, and if properly tuned it blocks short transient and loud inputs, unlikely voice, from generating output DMI parameter vectors. In order to operate, the VCI4DMI requires bot gates to be open (green), thus these should be disabled or properly tuned.

Figure 16

A default combination of DMI parameters can be optionally forced on the DMI, via the MATLAB and DMI front-end chain, when the energy gate is closed. The user can specify the time interval that takes to linearly glide from the last output vector of DMI parameter to the defaults, as detailed in Figure 17.

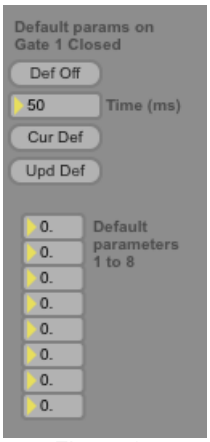

Figure 17

The VCI4DMI-Runtime.maxpat provides a simple method to trigger DMI sounds, generating MIDI notes from the voice input by tracking energy, pitch, and onset. Options and settings are shown in Figure 18. These voice characteristics are not used for the mapping to DMI real-valued parameters. The note pitch, velocity and note duration can be manually fixed or derived from the live voice tracking. The pitch can be filtered by user-defined scale selectable from a drop down menu. These are sent via MIDI to the instrument and the voice onset detection determines the transmission of the note-on message. These settings are stored in the structure containing the vocal gestural controller data.

Figure 18

All the changes made via the GUI to the options and settings of the system, vocal gestural controller, and DMI mapping are updated into the MATLAB bankV and bankD entries. The updated cell arrays of vocal gestural controller and DMI mappings are returned by the runtimeVCI4DMI MATLAB function when execution ends. To save these into two new MATLAB workspace variables run the command with two output arguments: `[bankVout,bankDout]=runtimeVCI4DMI(bankV,bankD)`. The cell arrays "bankVout" and "bankDout" will include individual entries with all options and settings selected during runtime.

For novice users, the most effective settings to tune the voice-to-instrument mapping, generated unsupervisedly, are:
- Out Inversion, Map Mode and Validity Region in Figure 14
- Map mode and Radius value (search mode must be in Radius) in Figure 15
- Features low pass filter in Figure 13.