

Manual for RealSong_control

Kosuke Hamaguchi

Contents

1: Introduction

2: Requirements for using RealSong_control

3: How to use RealSong_control

4. Known issues

1: Introduction

Overview of RealSong_control

RealSong_control is a data acquisition program run on MATLAB to conduct intracellular recordings in singing birds while distorting auditory feedback (DAF) of an animal by detecting a specific sound and broadcast pre-recorded sound. For this purpose, it has two operation modes. First mode is DAF mode in which the program continuously monitor microphone signal to detect of a specific sound features to triggers DAF and record up to 3 channel data. Second mode is cell search mode in which the program repeatedly inject current and monitor voltage response of the sharp electrode to detect a cell in front of the electrode.

This program is mainly used for sharp electrode recording in singing birds which receives DAF, but you can use this program without recording neural signal.

Acknowledgements

Kosuke Hamaguchi, Department of Neurobiology, Duke University Medical Center wrote this software. This software is strongly influenced by several other programs, but has tremendously benefitted from the following two. The sound feature calculation part is based on Segal Saar's MATLAB version of Sound Analysis Pro. The sound chunk detection was initially implemented by libsvm by Chih-Chung Chang and Chih-Jen Lin (<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>). This software was developed for research funded by NIH R01: DC02524 and NIH R21: NS079929.

Please cite the following publication when using the software:

Kosuke Hamaguchi, Katherine A. Tschida, Inho Yoon, Bruce R. Donald, and Richard Mooney (2014). Auditory synapses to song premotor neurons are gated off during vocalization, **eLife XX, XXXX**.

2: Requirements for using RealSong_control

2.1. Software requirements

The software is written for MATLAB 2012b or higher using Simulink, xPCtarget toolbox. Thus you need to install the followings.

- MATLAB 2012b or higher.
- Simulink
- xPCtarget toolbox.

It also requires C-compiler to compile Simulink model (which is freely available). Full details of installing and setting up xPCtarget environment can be found in the following Mathworks website (<http://www.mathworks.com/help/xpc/gs/setup-and-configuration.html>).

2.2 Hardware requirements

- Two PCs.

One PC is installed the above software and works as host PC. The other PC will be booted as target PC and installed a data acquisition board (DAQ board). Each PC needs to be connected to Ethernet to communicate with each other.

- DAQ board from National Instruments.
We used PCI-6251, PCI-6052, PCI-6070.

Analog inputs

- Connect microphone signal to the analog in terminal AI0. Data is sampled at 44.1kHz but downsampled to 22kHz when it is saved.
- Connect neural signal to the analog terminal AI1. Sampling rate is the same as above.
- If you use AxoClamp2B to control injection current to the pipette, and want to monitor the injection current level, connect Im output to AI2. Sampling rate is 1kHz when saved.


Analog outputs

- Connect speaker to analog output terminal AO0.
Speaker gain needs to be adjusted before use.
- Connect command current terminal to analog output AO1.
This is to control injection current during the cell search mode (Oscillo mode).

Digital outputs

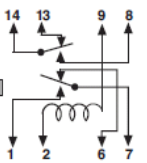
- To control third apparatus by using just two analog outputs, we use a relay switch to route output signals. This requires time sharing the output signal, therefore it cannot control 3

external apparatus simultaneously. We use DIO (Digital Output 0) to trigger relay switch. Here is the data sheet of the relay switch we used and connection diagram. You will not need these if you use this program just for DAF experiments.

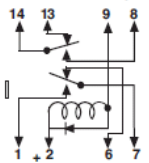


DPDT, 0.25 AMP			
W172DIP-17	5	46 Ω	540
W172DIP-19	12	266 Ω	540
W172DIP-20	24	1066 Ω	540
DPDT WITH CLAMPING DIODE, 0.25 AMP			
W172DIP-21	5	46 Ω	540
W172DIP-23	12	266 Ω	540
W172DIP-24	24	1066 Ω	540

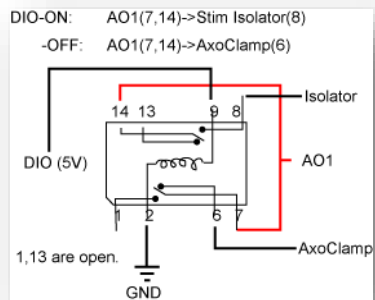
DPDT



DPDT WITH DIODE



SEE END OF SECTION 6 FOR CROSS REFERENCE



3. How to use RealSong_control

3.1) Setup

Setup host PC

3.1.1) Setup compiler. In MATLAB, type

```
>> xpcsetCC –setup
```

Chose compiler.

3.1.2) Setup target pc IP address (case for network bootable PC)

```
>> xpcexplr
```

TargetPC1->Configuration->Communication: select Host Target communication: TCP/IP

Set the following information

Target TPC/IP address: check them in target PC IP address (ex, 152.16.226.228,
use ipconfig /all in Dos-prompt to check this.

TCP/IP target port: 22222

LAN subnet mask address: 255.255.255.0

TCP/IP gateway address check this when you check TCP/IP address.

TCP/IP target driver: Auto (If target PC's ethernet driver is not supported, you will receive error later.)

3.1.3) Start network boot loader

TargetPC1->Configuration: Network boot panel: Press "Create Network Boot image" button.

You may be asked by windows whether to unblock the matlab xpcexplr servers. Unblock them.

Setup target PC (see also <http://www.mathworks.com/help/toolbox/xpc/gs/f1-4598.html>)

3.1.4) Restart target PC, go to BIOS setup mode.

3.1.5) Change boot sequence to make the network boot first

3.1.6) Disable USB and other unnecessary processes

3.1.7) then reboot

If network boot service is running in host PC, your target PC will show up a window telling it is booting.

Test target pc

```
>> xpctest
```

If no error occurs, then successfully host PC and target PC setup is done.

Bench mark your target PC

```
>> xpcbench('this')
```

IF you see "CPU Overload", it is likely that your PC has either one or some of PnP, USB, Advanced

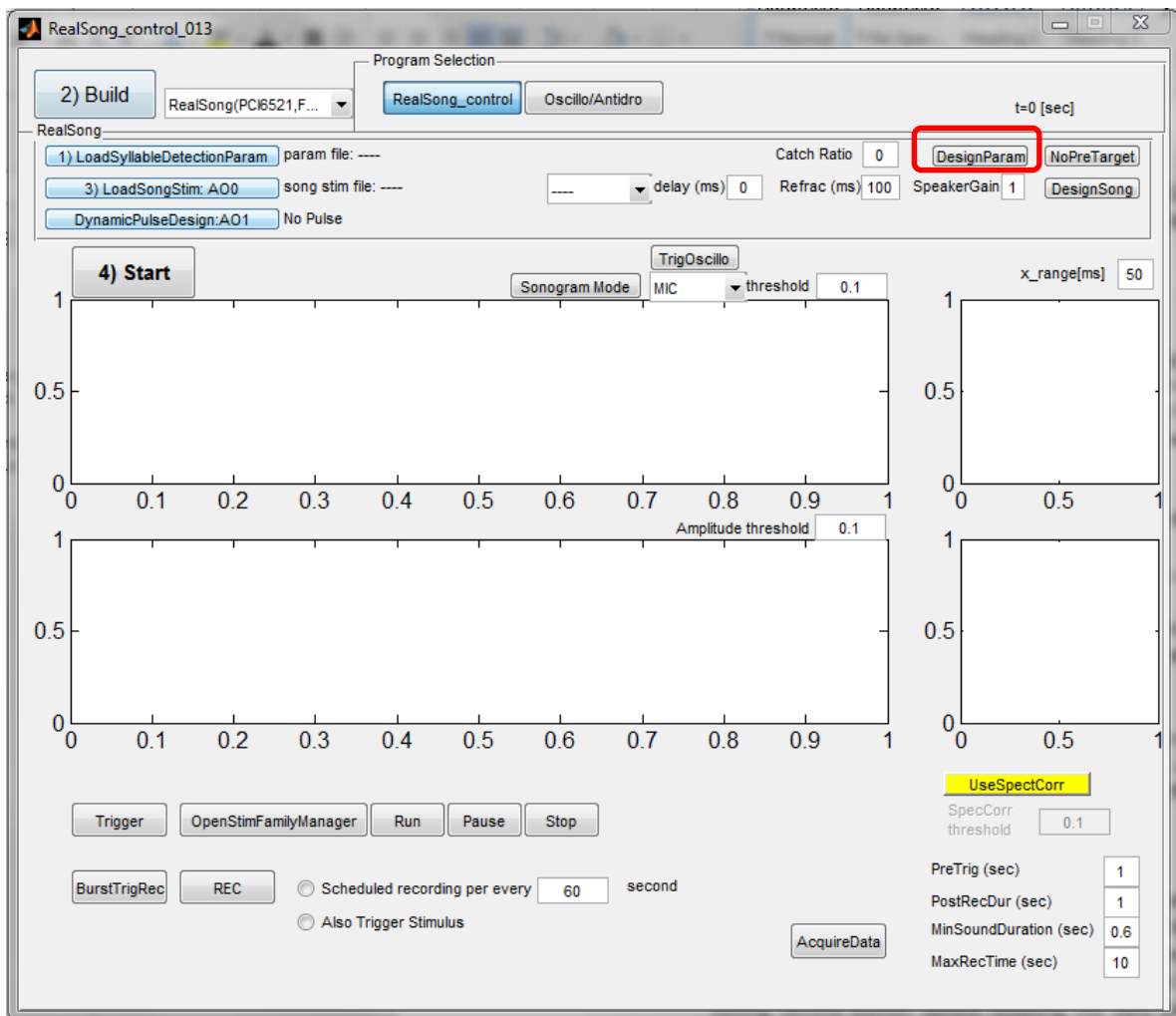
Power Management is enabled. They interrupt the CPU process and could cause "CPU Overload" error. You can disable them at BIOS setup. In my experience, you should definitely turn off USB.

3.2) Making template (define target syllable)

Type

```
>> RealSong_control_013
```

MATLAB will launch the following GUI.



Overview of sound detection in RealSong_control

The program continuously monitors the sound amplitude. When sound amplitude exceeds a certain threshold, it starts to calculate mean and variance of sound features (amplitude, entropy, FM, meanfreq, etc...) until sound level drops below the threshold again. At this point, the program knows the mean and variance of the sound features averaged during the sound duration. We classify sound chunks into two category by using Support Vector Machine (SVM). Once the program detected a specific sound chunk, it goes into sensitive period. The start and the end of sensitive period is defined from the **end of Pre target**. During this sensitive period, sound features are tested whether it satisfies a certain “hit” criteria at every 1ms. For example, you can set sound amplitude above 0.1 and less than Inf, and also set meanfreq more than 5000 Hz and less than 5100 Hz to hit a specific sound segments. If a sound segment satisfies all the conditions for more than or equal to 3 ms, it will trigger TTL or speaker output.

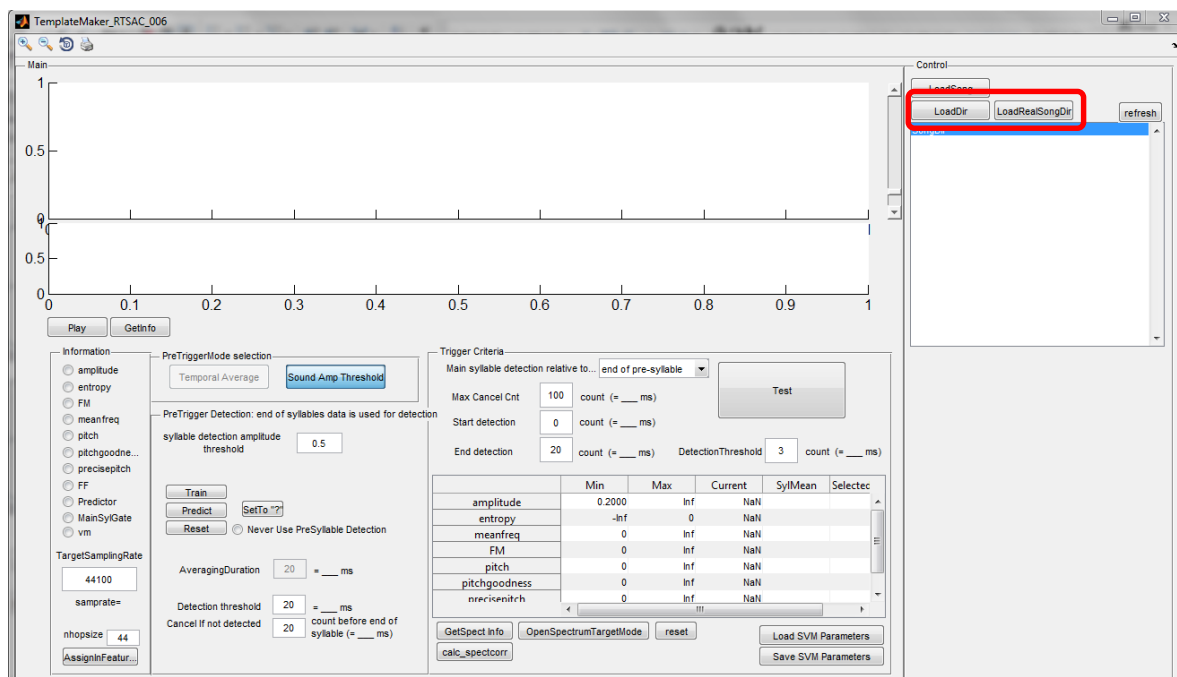
TemplateMaker_RTSAC_006 program is used to teach Support Vector Machine which one is the “Pre” target and set the hit criteria for the “Main” target.

To make a template for targeting a syllable, press “design template” button in RealSong_control GUI. It will show up the following screen. Please record your animal’s vocalization either using this program (recommended) or wav format.

RealSong data format : Press “LoadRealSongDir” button.

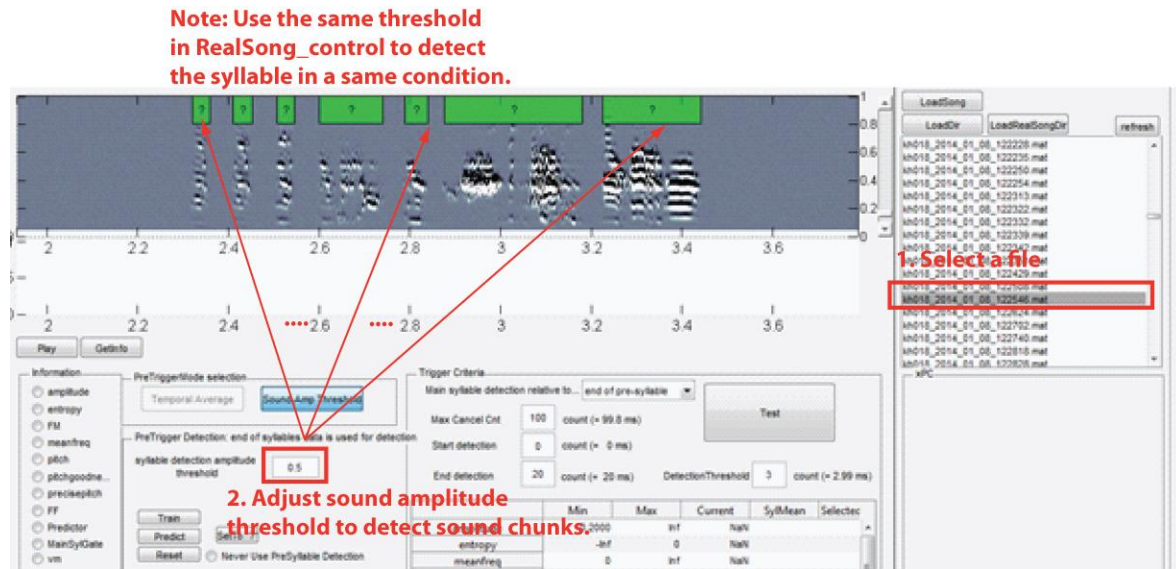
Wav data format : Press “LoadDir”

If this is the first time to run the program and do not have wav files, you can skip this and select “svm_demo” file as a target in next section.



Now, open a directory. Select a file that contains your target sound.

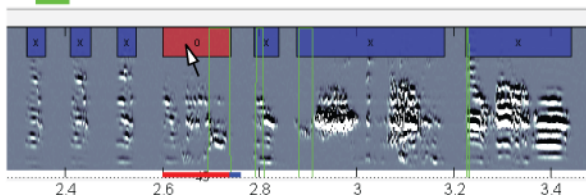
Adjust sound amplitude so that your “pre” target sound chunk is reliably detected in a same length between files.



Select “Pre” target syllable and “NOT” pre-target syllables by clicking the box.

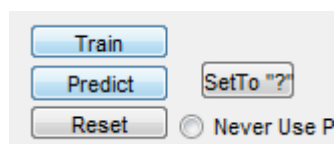
Right/Left click each box to cycle through

- o: target
 - x: not target
 - ?: non teacher data.
- } teacher data



Note: minimize the total number of teacher data (o and x) to avoid CPU overload error. Less than 20 is recommended.

Press “Train” to train SVM with your teacher data. “Predict” will show you the red bar and blue bar on the bottom.



The red bar is the detected “Pre” target. Blue bar is the sensitive period. Green thin lines is the SVM’s prediction. Up state of green line means SVM predicted that sound segment is your “Pre”

target. Sometimes SVM detect very short sound segment as “Pre” target even outside of your true “Pre” target. To prevent triggering sensitive period by this short time segment, we need to set a threshold to start sensitive period.

Detection threshold	<input type="text" value="20"/>	= 20ms
Cancel if not detected	<input type="text" value="20"/>	count before end of syllable (= 20ms)

Detection threshold defines this cutoff duration. For a sound chunk to be recognized as “Pre” target, SVM prediction needs to be in UP state at least more than 20ms (20 sound segments) in total (not necessarily continuous). *Detection threshold* can be shorter if your “Pre” target is actually shorter than 20ms.

Sometimes, a part of a sound segment in a non-target syllable can be very similar to your “Pre” target. In most of the case, mean and variance of features eventually deviate from your “Pre” target syllables’ mean and variance, near the end of the sound. *Cancel if not detected* parameter can filter out these syllables. It defines the duration that SVM needs to be in UP states at least once relative to the end of the syllable.

Choose different files and train SVM until you can reliably detect “Pre” target. However, be careful not to add too much “O” target, or “X” non-target. Just use “?” unknown as much as possible. The computation time linearly increases as the number of teacher data increases. In our experience, if we have more than 20 teacher data, it tends to cause “CPU” overload error.

Also, make sure to teach SVM that cage noise is not your target.

Set sensitive period



Once you are confident about “Pre” target detection, set the start and end time points of sensitive period (blue bar). Note that these parameters are defined relative to the end of “Pre” target. *Max Cancel Cnt* needs to be longer than *End* duration.

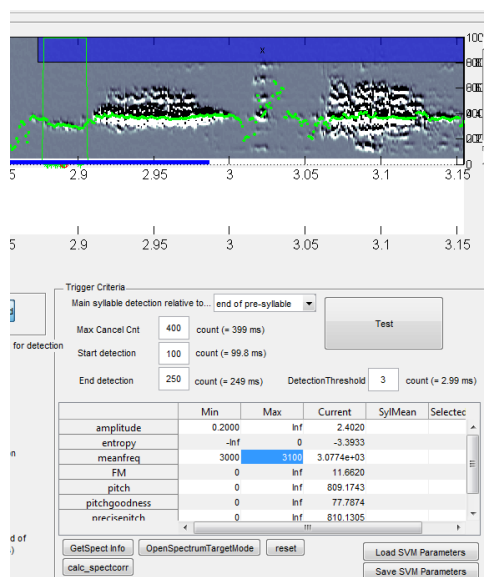
Press “Predict” to update the blue and red bar. You can see blue bar position changes.

Set “hit” criteria

Now we want to set criteria to hit “Main” target. The following example is the condition to hit any sound which sound amplitude is above 0.2. All the other parameters are set between [0 inf] or [-inf 0] to cover all the possible range.

	Min	Max	Current	SylMean	Selectec
amplitude	0.2000	Inf	NaN		
entropy	-Inf	0	NaN		
meanfreq	0	Inf	NaN		
FM	0	Inf	NaN		
pitch	0	Inf	NaN		
pitchgoodness	0	Inf	NaN		
nrcrisenitch	0	Inf	NaN		

To know the pitch or sound features of your “Main” target syllable, press “GetSpect Info”. The mouse arrow will turn into a cross-cursor, and allow you to select a sound segment you are interested in. In the next panel, we set meanfreq > 3000 and meanfreq < 3100, amplitude > 0.2.



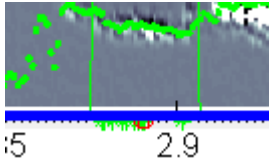
Then, press “Test” to check when the sound segments satisfies the “hit” criteria. You can press other sound feature radio button on the most left box to plot sound features (shown in here is *meanfreq*).

The sound segment that satisfies these conditions are shown in green circle on the bottom.

Final hit criteria imposed by this program is that sound segments continues to satisfy this hit criteria more than 3 ms. these timings are marked with red circle. This will be the actual trigger time point in the actual experiments.

*Currently, changing the parameter “Detection

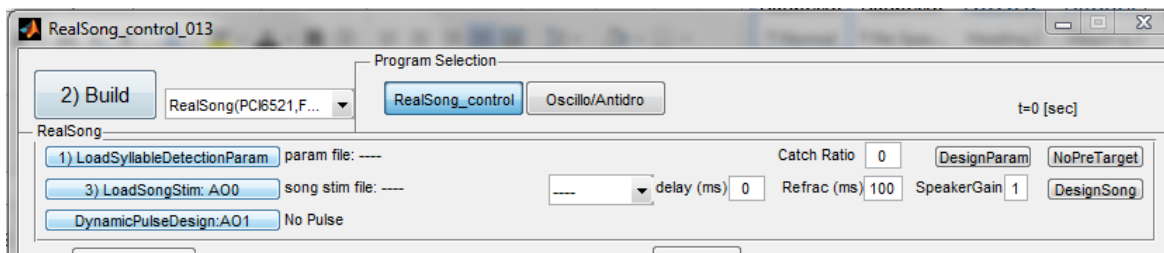
Threshold=3ms”, is not reflected in the RealSong_control program. Needs to be fixed.



Then, save the SVM data file by pressing “Save SVM Parameters”. You can load the previously saved data by pressing “Load SVM Parameters” after loading data directory.

3.3) Start RealSong_control

1) Load SVM data file in RealSong_control program by pressing “1) LoadSyllableDetectionParam” button.



2) Choose your DAQ board (PCI-6052, PCI-6070, PCI-6251) and mode (full mode, mic-only mode) from the dropdown list next to “2) Build” button.

- Full mode is 3-channel recording mode. It receives MIC (AI0), Vm (AI1), and injection current (AI2).
- Mic-only mode is single-channel (MIC) mode.

3) Build the model. Press “2) Build”. It may take one – two minutes for the initial compile.

4) Choose the playback stimuli.

5) Press “start” for start sound-event driven recording.

Select file name. RealSong_control will attach a suffix to the filename (in the date format, yyyy-mm-dd-hh-mm) and save the data when any sound event longer than 800ms occurs. You will be also asked the bias and gain for plotting. These settings are just for plotting purpose and do not change actual data.

Note that, RealSong_control can generate files which are slightly overlapped. You will need to eliminate duplicated data points when you batch process these files.

5) If you want to drive TTL, turn on “dynamic pulse”.

Important! Refractoriness starts to count from the beginning of the TTL. (This needs to be fixed).

Therefore, if your pulse duration is 100ms and refractoriness is 50ms, the duration of the TTL is limited to 50ms.

Set adequate level of sound amplitude threshold for detecting sound event (around x5 above baseline, but template detection also depends on this. Choose same level used in template making part).

6) Set catch ratio (0 is all hit, 1 is all catch)

7) Set audio speaker gain, press “trigger” to check the outputs.

You can also record continuously by pressing “REC”, or make it scheduled by clicking the radio button of “scheduled recording” and setting intervals. As a default, one recording file contains 10sec of data.

Data structure

The data file generated by RealSong_control contains two variables; parameter (param) and data (cbdata).

Easiest way to extract data is using the following command.

```
>> Data=realsong_dataclass(filename)
```

Data.t : Recorded time from the "Start".

Data.MIC; Input to AI0, Microphone

Data.Vm; Input to AI1. Converted to membrane potential in milli-volt, assuming that channel 1 has gain 10 (connected to 10Vm of AxoClamp).

Data.Im; Input to AI2.

Data.Speaker; Output of AO0.

Data.TTL; Output of AO1.

4) Known Issues

CPU overload error

This happens when target PC cannot finish their command within pre-defined time. There are many potential cause of this. Here are what we encountered during the course of experiments.

- 0) Using too many teacher data in SVM. -> reduce the number of teacher data.
- 1) Lack of CPU power in target PC. -> upgrade target PC.
- 2) Interruption at BIOS level might be the cause. Make sure USB and power management are turned off at BIOS level in target PC.
- 3) Slow Ethernet connection. Changing Ethernet card on either master or slave side worked once.
- 4) Changing compiler also works.
- 5) If this happens right after channel selection dialogue, please wait for 3 sec before pressing "OK".

The program freeze or does not generate expected behavior when I press multiple buttons quickly.

Please do not press buttons sequentially before you see any expected changes. MATLAB GUI triggers a callback function when it is clicked. The callback function try to access to the variable when it is available. It does not wait for finishing the previous Callback function's process. Therefore, hitting second button before finishing a process triggered by first button could have unexpected consequence.

There must be a setting to change this behavior, but current solution is to wait for pressing any button until you see any expected change. Please let me know if you know the correct way to fix this.