MiniScan Manual for fUS data acquisition and processing

from the article “Whole-brain functional ultrasound imaging in awake head-fixed mice”
by Clément Brunner[1,2,3,4,*], Micheline Grillet[1,2,3,4,*], Alan Urban[1,2,3,4], Botond Roska[5,6,7], Gabriel Montaldo[1,2,3,4,#] & Emilie Macé[8,#]
Nature Protocols, 2021

[1] Neuro-Electronics Research Flanders, Leuven, Belgium.
[2] VIB, Leuven, Belgium.
[3] Imec, Leuven, Belgium.
[4] Department of neurosciences, KU Leuven, Leuven, Belgium.
[5] Institute of Molecular and Clinical Ophthalmology Basel, Basel, Switzerland.
[6] University of Basel, Basel, Switzerland.
[7] NCCR Molecular Systems Engineering, Basel, Switzerland
[8] Brain-Wide Circuits for Behavior Research Group, Max Planck Institute of Neurobiology, Martinsried, Germany.

* These authors contributed equally to this work
[#] These authors share the senior authorship

Correspondence should be addressed to E.M. (emace@neuro.mpg.de) and G.M. (gabriel.montaldo@nerf.be).

Link to article: https://www.nature.com/articles/s41596-021-00548-8
DOI: https://doi.org/10.1038/s41596-021-00548-8

See Github repository for potential updates: https://github.com/nerf-common/whole-brain-fUS
Download the dataset in the Zenodo repository: http://doi.org/10.5281/zenodo.4905862

miniScan acquisition system

The miniScan is an acquisition system suitable for a first approach to the functional ultrasound imaging method. With a simple hardware implementation, it enables to run dedicated sequences for acquisition. Moreover, it includes a software package to register the functional ultrasound imaging data with the Allen Mouse Common Coordinate Framework, v3 (CCF) and compute both functional activity map and time-course activity of individual regions.

1. Installation.

Materials. The hardware is composed of the following parts:

- Ultrafast ultrasound scanner (Vantage 128 or 256, Verasonics) running versions 3.0.7, 3.4.0, 3.4.2, or 3.4.3. Both MATLAB and Verasonics software are supposed to be installed in the master computer provided with the Vantage system.
- Graphic processing unit (Nvidia, GeForce). Recommended models are GTX 1060/1070/1080/1080-Ti or RTX series.

- Motorized-linear stage (T-LSM100A, Zaber Technologies Inc.). See Note 1 to use another motor.
- Linear ultrasound transducer (L15-Xtech, Vermon)

The plans of the probe holder and the headpost are included in software package.

Installation of the fUS system.

- Plug the Vantage system to the master computer by following instructions given by the "Vantage Initial Setup" documentation.
  Run the `SystemVerificationTest.m` to ensure that the system works correctly with a valid license.
- Switch off the master computer, open the cover and remove the graphic card unit.
- Insert the Nvdia GeForce card in the PCIe slot and plug the power supply of the card.
- Connect the motorized linear stage to the serial port COM1 of the master computer

Software installation.

- Download the CUDA toolkits and follow installer guidelines.
- Download and unzip '`miniScan.zip`' (see Supplementary Software) into the root of the C:\ drive of the master computer. Then, found 3 folders under `C:\miniScan`:

`\acquisition`	contains the acquisition software,
`\data`	the folder where the acquired data is saved,
`\miscellaneous`	contains both test and stimulus codes.
`\mechanic_parts`	plans of the probe holder and the mouse head post

Verification of installed components.

- Launch MATLAB and select the path `C:\miniScan\miscellaneous`
- Test the GPU computing by running:

```
>> test01_ComputingGPU        <enter>
```

- The computing time is controlled through a simulated data.
- Check both motor communication and function as follow:

```
>> test02_Motor      <enter>
```

The motor will move 1 cm back and forth.

! CAUTION Free-up space around the motor to avoid damages of neighboring systems or tools. If the ultrasound probe is already mounted on the motorized-linear stage, we strongly advise users to free-up space around it to avoid damaging the probe.

miniScan setup.

Inside `C:\miniScan`, edit `start_miniScan.m` as follow:

- line 5, select the path to the Verasonics software,
- line 7, select the version of the Verasonics software by changing the string variable: `VANTAGEVERSION`. The '`Emulator`' version that enables to run `miniScan` without the Verasonics hardware/software.

Note 1: Use of a different motor.

To use a different motor, users must program a MATLAB object with the template `stpMotorTemplate.m` located in `C:\miniScan\miscellaneous`. After programming and testing the action of each method, rename the class to `stpMotor` and replace it in `C:\miniScan\miniScan`

2. Using miniScan.

In MATLAB, select the folder `C:\miniScan` and run:

```
>> start_miniScan        <enter>
```

A 3-panel graphic user interface opens as shown in Figure 1.a. The left panel allows to adjust the settings, the middle one allows to set acquisition modalities and the right panel contains the experimental notebook.

2.1. Settings panel.

The panel ‘Settings’ allows to fill in the acquisition parameters in 4 distinct sections:

- In the ‘File name’ section enter the 'mouseID'. This will tag all acquisitions along the imaging sessions.
- In the 'Sequence' section, click the 'Load' button. The sequence will be loaded to the Verasonics system. Once the Acquisition status box turns ‘Free’ and green, the system is ready to proceed with the next step.
- In the 'Motor’ section, click the 'Open’ button (change the name of the serial port if your motor is not connected to COM1).
- In the 'Stimulus' section, select the serial port connected to the stimulus computer and click the 'Open’ button. This port allows the synchronization to other devices with default MATLAB parameters (9600 bauds, 8 data bits, parity none, terminator LF).

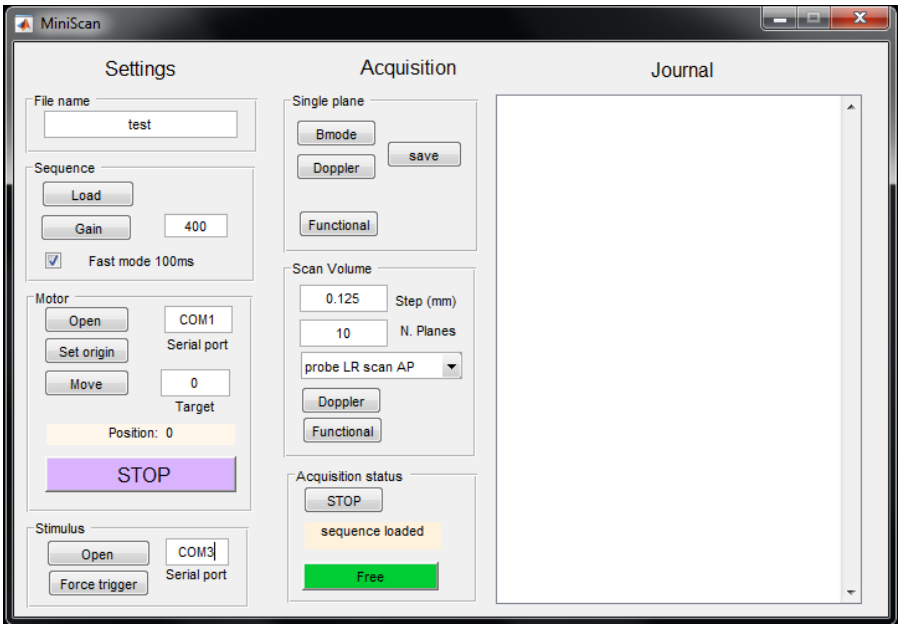


Figure 1. Graphical user interphase of the MiniScan acquisition software.

Using the motor. The motor can be moved using the manual knob located at its back or through the GUI by setting the position ('Target' field) and moving the probe ('Move' button). Distances are in mm. Set a new origin (i.e., position 0) by clicking the 'Set origin' button, the displacement of the probe is then measured from this new origin position. To avoid any issue with the transducer while in motion, we strongly recommend testing the motor in advance and identify the positive and negative directions of the motor. To interrupt the movement of the motor, hit the 'STOP' button.

Forcer trigger. The visual stimulus is triggered by a character 'T' sent through a serial port. To test the visual stimulus, trig it manually by clicking the 'Force Trigger' button**.**

Fast mode 100ms. The Default Mode offers a temporal resolution of 500ms. One can select a faster imaging mode with a 100-ms temporal resolution by selecting the 'Fast mode 100ms' checkbox in the 'Sequence' section. This mode disables the real-time display of µDoppler images to save computing resources.

Gain. The 'Gain' button of the 'Sequence' section adapts the gain in the reception amplifier. The gain value ranges from 1 to 1000. If the Doppler image display shows some saturation reduce the gain value.

2.2. Acquisition panel.

Single plane.

The 'Single image' subpanel enables to freely explore the brain volume with single plane acquisitions.

The 'Bmode' button starts the real-time visualization of rough echo images of the tissue. While this mode is not essential for functional imaging, it remains highly useful to align the probe to the brain, detect trapped air bubbles or find the edges of the cranial window. Stop the live imaging by clicking a second time to the 'Bmode' button.

The 'Doppler' button allows the acquisition of a single high-quality Doppler image in 2 seconds. This mode is used to check the status of the brain tissue, perfusion and image quality.

The 'Save' button saves the last image computed in Matlab format. This file is named after the 'File name' previously filled in the 'Settings' panel, an extension provides the type of image and a 'hhmmss' timestamp (e.g., `mouseID_110139_DoppPlane.mat`). The saved file is then logged on to the 'Journal' where extra information can be edited.

The 'Functional' button to perform a functional acquisition composed of 70 µDoppler images at 2 Hz (10 Hz in fast mode). At image 35, the master computer automatically triggers the stimuli. Once completed, the file is automatically saved both timestamp and 'FusPlane' extensions (e.g., `mouseID_110148_FusPlane.mat`). See Appendix 2 for more information about the ultrasound sequence.

Scan volume.
Before addressing functional activity of a brain volume, set (i) step size information (in mm) in the 'Step' field and (ii) the number of planes in the 'N.planes' field. The scan starts at the position 0 of the motor and moves with the defined step.

For the registration with the atlas it is important to indicate the orientation of the probe and the direction of scanning. 8 possibilities are possible in the list menu

Probe position
LR: coronal section, first element at Left
RL: coronal section, first element at Right
AP: sagittal section, first element is Anterior
PA: sagittal section, first element is Posterior

Scan direction of the motor
LR: scanning from Left to Right
RL: scanning from Right to left
AP: scanning from Anterior to Posterior
PA: scanning form Posterior to Anterior

! CAUTION Before starting the scanning session, confirm that the 'Set origin' has been performed. Thus, we strongly recommend moving the motor to the origin to avoid unexpected movements from a wrong origin.

Doppler volume. The 'Doppler' button performs an anatomical scan consisting on the acquisition of single µDoppler images of each plane of the scanned volume. This scan is used for the registration with the Allen Mouse CCF and can be acquired at higher spatial resolution (~0.1mm) than the functional scan. The Doppler volume is automatically saved with both timestamp and 'AnatVol' extensions (e.g., `mouseID_111201_AnatVol.mat`).

Functional volume. The 'Functional' button performs a functional acquisition composed of 70 µDoppler images at 2Hz (or 10Hz using the fast mode) for each plane of the scan. At image 35, the master computer automatically triggers the stimulus by sending a character 'T' trough the stimulus serial port. Once completed, the scan composed of 'Nplanes' is automatically saved with both timestamp and 'FusVol' extensions (i.e., `mouseID_111621_FusVol.mat`). See Appendix 2 for more information about the ultrasound sequence.

2.3. Acquisition status and Journal subpanels.

Acquisition status. This subpanel displays real-time information of the acquisition process including the number of frames acquired and the frame rate. Furthermore, the

acquisition can be aborted at any moment by hitting the 'STOP' button, the acquired data until this point will be saved as described above.

Journal. The 'Journal' is a text file used to log all the files recorded over sessions. This section can be complemented with experimental information. The file is automatically updated after each modification and saved in .txt format.

Images, scans and journal files are all saved in `C:\miniScan\data`.

3. Using customer parameters.

The miniScan provides a default set of parameters optimized for the probe L15-Xtech and the experimental configuration described in the protocol. This default configuration constitutes a basic example to start with the fUS method but most of the users will need some adaptations for their specific experiments. This section is dedicated to users interested in adapting the acquisition and imaging parameters to their specific applications.

3.1 Changing the acquisition parameters.

The system starts with a default configuration defined in the file `parametersDefault.m` (folder `c:\miniScan\acquisition`). Users can modify the file `parametersUser.m` in the folder `c:\miniScan\acquisition` and create new parameters files adapted for different experimental conditions. To modify one or multiple parameters you need to perform the following steps:

1) Edit the file `parametersUser.m` and modify the desired parameter. For example, change `par.imageOriginZ=6`. This parameter changes the starting depth of the image to 6mm.
2) Start the miniScan system.
3) To load the parameters using the method `loadUserParameters` with the file name as argument

```
>> SCAN.loadUserParameters('parametersUser');
```

You can check that the image starts at 6mm instead of 3mm as in the default configuration. With this method, different users can save the configuration in different file names.

Some parameters perform simple modifications and are easy to manage. Some others need a deeper understanding of the ultrasound system and must be modified carefully. For example, an increase in the number of angles can be useful to increase the resolution but it implicates more data transfer and computing and the effective frame rate can be reduced.

To ease the adaptation, the parameters are organized in 8 categories that modify:

1) the transducer,
2) the ultrafast plane wave compound,
3) the beamforming parameters the imaged region,
4) the temporal resolution and acquisition duration,
5) the vascular filtering to select the blood signal,
6) the trigger out to synchronize with other instruments,
7) the mechanical scanning method,
8) other specific parameters of the electronics,

WARNING. The following 5 parameters modify the power emitted by the probe:

- `compoundAngles`
- `compoundFrameRate`
- `compoundAverage`
- `emissionLenght`
- `HVset`

High emission power can destroy the probe. Be **extremely careful** with a modification of **HVset**. The electronics can send stimulus pulses until 100V that can destroy the probe. As a protection, the parameters file has a basic check of the emission power between lines 68 to 79. We considerate a limit of 1% of emission duty cycle and a maximal voltage of 20V. This limit is acceptable for the probe L15-Xtech but if the user changes the probe, he must carefully check the new limits. In general, higher is the frequency, lower is the maximum voltage. We discharged from all responsibility in case of probe damages.

3.2 Change the ultrasound probe.

The following parameters enable to modify the ultrasound probe.

```
% transducer parameters
par.probeFreq=15.265;           % central frequency of the probe (MHz)
par.probeElementSize=0.1;       % size of the probe elements (mm)
par.probeNumberElements=128;    % number of elements  of the probe.
par.antiAliasCutoff=30;         % antialiasing filter (MHz)
par.LnaZinSel=25;               % amplifier impedance (integer 0 to 31)
par.HVset=25;                   % WARNING POWER stimulation voltage.
```

`probeFreq` is the central frequency of the probe in MHz. This frequency must be one of the demodulation frequencies enabled by the reception electronics (See table 3.3.1.1 of the vantage programming manual). Accepted frequencies in the range of 5 to 15 MHz are: 5.2, 5.6818, 6.25, 6.9444, 7.8125, 8.9286, 10.4167, 12.5, 15.625.

`probeElementSize` is the distance between two elements of the probe (pitch) in mm.

`probeNumberElements` is the number of elements of the probe. If the probe has less than 128 elements, the unused channels are disabled.

antiAliasCutoff defines the cutoff frequency of the antialiasing filter of the reception electronics. Accepted values (in MHz) are 5, 10, 15, 20 and 30. In general, this parameter must be approximately twice the central frequency of the probe.

LanZinSel defines the input impedance of the low noise amplifier (integer 0 to 31, see the verasonics manual).

HVset defines the stimulation voltage. WARNING: The electronics can send up to 100V that can potentially destroy the probe. Be very careful with this voltage.

3.3 Modify the ultrafast plane compound method.

These 5 parameters enable to modify the compound plane wave emission. See the appendix 2 Ultrasound sequence before modifying these parameters.

```
% compound definition
par.compoundAngles=[-6 -4 -2 0.25 2 4 6];  %WARNING POWER angles for…
par.compoundFrameRate=500;                 %WARNING POWER frame rate…
par.compoundAverage=3;                     %WARNING POWER number of averaging
par.emissionLenght=2;                      %WARNING POWER number of …
par.compoundAverageCoef=0.5;               %multiplicative coefficient…
```

compoundAngles is a vector to indicate the angles of the plane waves for the compound method in degrees. Increasing the number and the value of the angles can improve the contrast and resolution of the image at the cost of some extra computing time.

compoundAverage indicates the number of averages of the same plane wave emission. To increase the SNR, the same angle is emitted multiple times and the received signals are automatically added inside the electronics. As this averaging is done inside the electronics, the data transmitted to the computer and the computing time are not changed.

compoundAverageCoef is a multiplicative coefficient between 0 to 1 that is applied to the received echoes. The objective of this coefficient is to avoid a saturation of the reception when using a `compoundAverage`. The signals are digitized at 12 bits and, after the internal filtering, the output has 14 or 15 bits (depending on the filters) and they are stocked in standard integers of 16 bits. Adding 3 or more signals can produce an overflow of the 16 bits and unexpected artifacts in the image. To avoid this potential overflow, the `compoundAverageCoef` is recommended to be `1/compoundAverage`. In some cases (low signal), it can be `1/sqrt(compoundAverage)` to keep the maximal number of bits.

compoundFrameRate is the frame rate in Hz of the compound images. The firing rate (i.e. the frequency between two emissions) is `firingRate=length(compoundAngles)*compoundAverage*compoundFrameRate`.
It is important to check that the `firingRate` is physically possible. Firstly, this frequency cannot be higher than ~30KHz due to electronic restrictions. Secondly, the maximal depth that can be imaged depends of the `firingRate` as

`MaximalDepth=soundSpeed/(firingRate*2)`. This depth is the physical distance that the wave can travel and being reflected during two consecutive emissions.
The parameters program checks the consistency of the firing rate and displays a warning in case of error.

3.4 Modify the beamforming and imaging regions.

The beamforming can be parametrized using the following parameters

```
% imaging region and beamforming
par.imageOriginX=0;            % first position in depth dimension (mm)
par.imageOriginZ=3;            % first position in the lateral dimension(mm)
par.imagePixelX=0.1;           % pixel dimension in the lateral dimension (mm)
par.imagePixelZ=0.075;         % pixel dimension in the depth dimension (mm)
par.imageNumberPixelsX=128;    % number of pixels in lateral ( columns)
par.imageNumberPixelsZ=128;    % number of pixels in depth (rows)
par.imageFNumber=1;            % numerical aperture of the probe.
par.imageApodization=0.5;      % beamforming apodization (0 to 1)
par.soundSpeed=1.5;            % sound speed (mm/μs)
```

`imageOriginX`, **`imageOriginZ`**, **`imagePixelX`**, **`imagePixelZ`**, **`imageNumberPixelsX`**, **`imageNumberPixelsZ`** enable to define the region to image and the spatial resolution. The units of the first 4 parameters are mm. Figure 1 indicates the effect of each parameter. The absolute origin is defined at the surface of the element 0 (first channel) of the ultrasound probe. From this point, we designate the origin of the imaged region using `imageOriginX` and `imageOriginZ`. The size of the pixel is characterized by `imagePixelX` and `imagePixelZ` and the number of columns and rows to image by `imageNumberPixelsX` and `imageNumberPixelsZ` respectively.
Note that the number of columns is independent of the number of elements in the probe and the pixel size is also independent of the probe pitch. For example, for a probe of 128 elements and 0.1mm pitch it is legal to beamform 200 pixels of 0.05mm in the lateral dimension.

`imageFNumber` defines the f-number to perform the beamforming. The f-number describes the aperture over the focal distance, as a rule of thumb it is approximately the pitch of the probe divided by the wavelength.

`imageApodization` is a number between 0 and 1 that indicates the fraction of the channels that are apodizated in the beamforming. This parameter can be used in combination with imageFNumber to optimize the resolution of the image.

`soundSpeed` determines the sound speed of the medium in mm/μs.

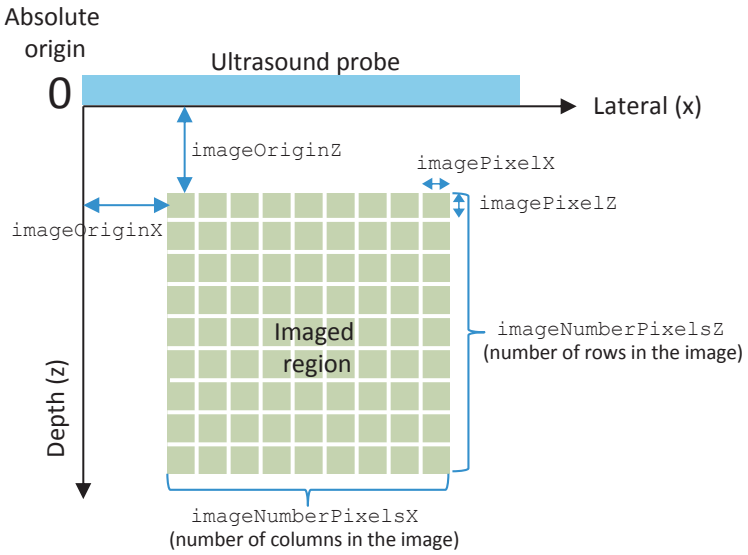


Figure 2. Effect of the beamforming parameters in the imaged region.

Changing the vascular filter.

A high pass filter and a singular value decomposition filter are implemented to select the blood signal and reject the tissue and movement signal. They can be set up with the following parameters.

```
% vascular filtering
par.vascularFilterSVD=0.1;        % fraction of singular values to eliminate.
par.vascularFilterHighPass=15;    % high pass filter cutoff (Hz).
```

`vascularFilterHighPass` is the cutoff frequency of a high pass filter. If the value is 0, the high pass filter is bypassed.

`vascularFilterSVD` is a number between 0 and 1 that indicates the fraction of singular vectors to be eliminated in the filter. The filter can be adjusted depending on the experimental configuration. If the movements are moderate (for example in anesthetized animals), this filter can be relatively low enabling to select more signal in the low frequencies associated to the smallest vessels. On the contrary, when using awake and active animals, it is recommended to increase the value of the filter.

3.5 Temporal resolution and imaging duration.

Two parameters enable to control the temporal resolution and the number of vascular images to acquire:

```
% temporal resolution
par.temporalResolutionNormal=5;  % temporal resolution in units of 0.1s
par.numberVascularImages=70;     % total number of images in a fUS acquisition
```

`temporalResolutionNormal` is an integer number. In the normal mode, this parameter fixes the temporal resolution as

dT = temporalResolutionNormal/compoundFrameRate*50.

Using the default compoundFrameRate of 500Hz the temporal resolution is

0.1s*temporalResolutionNormal.

The temporal resolution is equivalent to the “exposure time” in an optical camera. The vascular image is computed as the average intensity of the filtered compound images. This parameter changes the number of compound images to average as a multiple of 50 images in the present implementation. However, it is important to understand that using the maximal speed (i.e the fast mode or temporalResolutionNormal=1) does not reduce the SNR. For example, the user can (in post-processing) average 5 fast vascular images and obtain the same SNR than averaging inside the system by using a temporalResolutionNormal of 5. The main effect of this parameter is a reduction in the volume of data to save.

`numberVascularImages` is the total number of vascular images in a fUS trial.

3.6 Configure trigger.

The trigger can be configured with the following parameters:

```
% trigger
par.serialTriggerOut=[10 20 30];  % images to send a serial trigger.
par.electronicTriggerOut=0;       % sets the trigger out (0 off, 1 on).
```

`serialTriggerOut` Number of images to send a trigger (a character ‘t’) by the serial port. It can be a vector with multiple images, in the example the trigger is activated at images 10, 20 and 30.

`electronicTriggerOut` must be 0 or 1. If 1, trigger signals are sent by the trigger out connector placed in the back panel of the Vantage electronics. This signal is repeated each 50 compound images. This signal can be used to synchronize different tools running in asynchronous mode (cameras, electrophysiological recordings, …).

3.7 Mechanical scanning.

These parameters enable to scan the brain planes with a list of preselected positions.

```
% motor
par.motorRandom=0;                          % 1=random mode, 0=linear scan
par.motorRandomPositions=[0.25 0.75 0.5];   % positions for random motor.
```

`motorRandom`. can be 0 or 1. If 0, the scan is linear. If 1, the scan uses the random positions defined in motorRandomPositions.

`motorRandomPositions` defines a list of positions when motorRandom is 1. The brain is scanned at the positions defined in this vector. The number of defined positions must be equal or higher than the number of scanning planes.

Note: these two parameters are independent of the ultrasound part of the system and can be changed directly in the properties of the SCAN object. For example, to start a random scanning, you can perform:

```
SCAN.motorRandom=1;
SCAN.motorRandomPositions=[0 1.5 0.5 3.0 2.5 1.0 2.0 0.75];
```

By this method, the user can change the positions or switch between random/linear at each new scan without reloading all the sequence parameters.

Specific parameters of the electronics.

These last set of parameters control some specific parameters of the electronics:

```
% other specific parameters of the electronics
par.DebugTiming=0;      % 0 or 1.  when 1 there is a warning each time the…
par.codeSampleMode=2;   % must be 2 or 4. 2 sets the acquisition mode
par.InputFilter=[+0.00058 +0.00018 -0.00113 -0.00128 -0.00119 +0.00656 …
                        % 21 coefficients of the input for filter.
```

`DebugTiming` must be 0 or 1. If 1, the system displays a message each time the sequencer cannot reach the specified timing. If the computing is not fast enough to process the data in real time the sequence must be stopped for some time waiting for the processing to finish. This "lost time" is displayed by the system. The lost time has no effect in an individual image because the system stops during two images however it extends the acquisition time of the trial.
This flag is used during the debug of a new set of parameters to verify if the computing can be performed in real time.

`codeSamplingMode` must be 2 or 4. The value 2 selects the acquisition mode 'mode100BW' and the value 4 the 'mode200BW' (see Vantage manual section 3.3.1 parameter `sampleMode`). The mode 200BW acquires 4 samples by wavelength, is the most generally used but consumes more memory and transfer time. The mode 100BW decimates the data by 2 at the cost of a minimal reduction in the bandwidth.

`inputFilter` is a set of 21 coefficients of the input filter (see Vantage manual section 3.3.1 receive object, `InputFilter`). The input filter must be adapted depending on the acquisition mode and the transducer bandwidth.

Processing software package

4.1. Installation.

Download and unzip '`processing.zip`' (see Supplementary Software) into the folder `C:\processing`. This folder is divided in 3 folders

`\codes`,	contains the user functions for data analysis.
`\examples`,	provides the examples for each function.
`\dataSamples`,	contains a set of data to explore the functions.

The processing package is independent of the acquisition hardware and can be copied and run in any computer with MATLAB 2015 or up. The dataSamples must be download in http://doi.org/10.5281/zenodo.4905862

! CAUTION Note that this data set is different than the version 1 to include the arbitrary scanning direction.

4.2. Data fus-structure format.

We defined a fus-structure as a MATLAB structure containing this 4 fields:

- `Type,` a string with 4 possibilities: 'plane', 'volume', 'fusplane' or 'fusvolume',
- `Data,` a multidimensional matrix of float (single or double),
- `VoxelSize,` gives the voxel size of each dimension in µm,
- `Planes,` provides the position of the motor for each acquired plane.
- `Direction,` is the orientation and direction of the 3 axis

The `miniScan` software saves all the data as fus-structures. The Table 1 summarizes the dimension and size of the fields for each of the 4 possible types.

Type	Plane	fusplane	Volume	fusvolume
Data dimensions	2	3	3	4
Data size	nz, nx	nz, nx, nt	nz, nx, ny	nz, nx, ny, nt
VoxelSize content	dz, dx	dz, dx	dz, dx, dy	dz,dx,dy
Planes size	1	1	Ny	ny
Acquisition mode	Bmode, Doppler	fUS single plane	Doppler volume	fUS volume

Table 1. nx, ny, nz are the number of points in the x, y, z directions respectively. Nt is the number of time points, dx, dy, dz are dimensions in µm in the x, y, z directions. Axis conventions are shown in Figure 1.b.

4.3. Processing functions.

To explore the processing function, move to the 'examples' folder and add a path to the codes and data examples,

```
>> cd C:\processing\examples  <enter>
>> addpath_codes_examples     <enter>
```

The examples 01 to 08 follows the order of the processing protocol. The processing functions are presented in chronological order following the protocol.

registrationccf.m opens a graphic user interface to register the anatomical scan with the Allen Mouse CCF, see Figure 3.

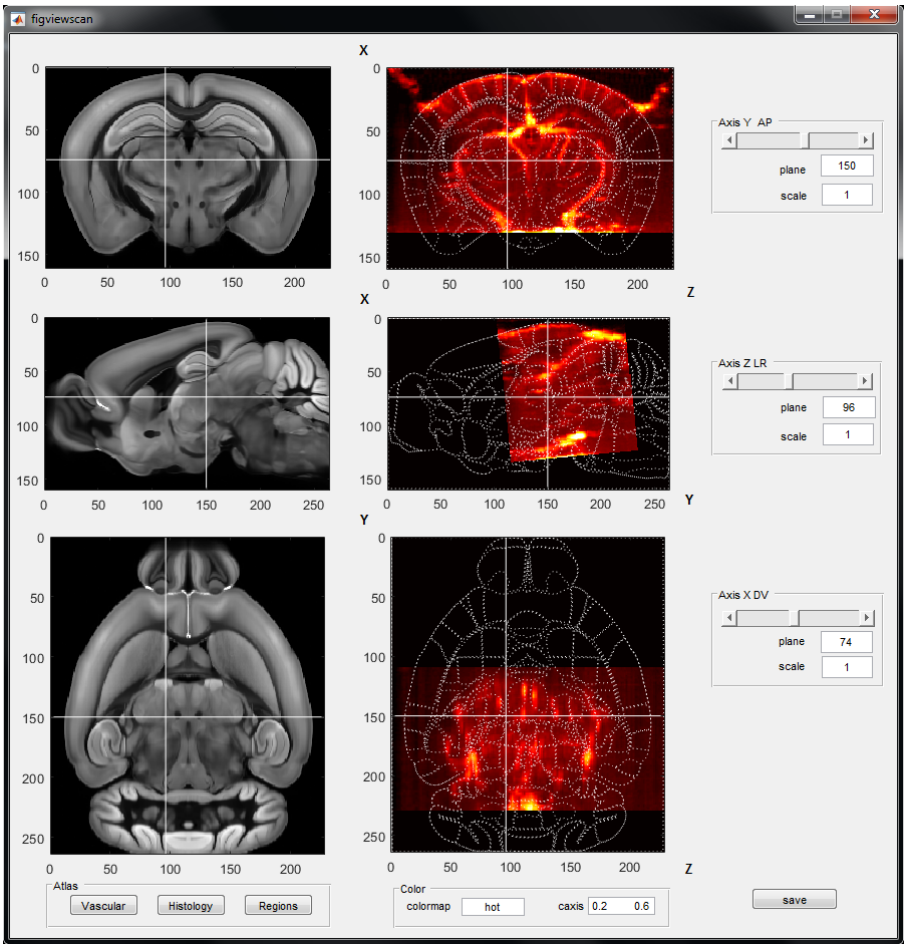


Figure 3. Graphical user interphase for data registering with the Allen mouse CCF atlas.

Syntax:

`registrationccf(atlas,anatomic)`
`registrationccf(atlas,anatomic,Transf)`

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,
`anatomic`, a fus-strucure of type volume containing the anatomical scan,
`Transf`, (optional) Affine transformation structure obtained from a previous execution of `registrationccf`. If this parameter is present, the data is transformed with the transformation `Transf` before starting the manual registration.

Output:

A structure `Transf` is saved in a file `Transformation.mat`. This structure contains the affine transformation matrix (field `M`) and the view size (field `size`).

Example:

`example01_registering.m`

Note:

The registration is done using an affine transformation (translation, rotation and scaling). To rotate and translate the anatomical volume, click to the μDoppler image to activate the image for rotation and translation (the arrow

pointer turns to a hand-pointer). Translate the volume by dragging the μDoppler image with the left-click of the mouse or rotate it with the right-click. The scaling is performed through a scaling coefficient in the x, y or z axis. Save the transformation matrix in the file '`Transformation.mat`' by hitting the 'Save' button.

dataStability.m. The animal movements may induce motion artifacts with an abrupt increase of signal. This function displays the distribution of the average value in the images, adjust a gaussian curve and classify as outliers all the images with an average value higher than 3 times the sigma of the gaussian curve.

Syntax:

`[outliers]=dataStability(scanfus)`

Inputs:

`scanfus`, fus-structure of type fusvolume.

Output:

`Outliers`. It is a 2D matix (ny by nt) of binary values indicating the position of the outliers in the ny planes and the nt time frames, 1 is an outlier and 0 is accepted frame. This output is used in the function `imageRejection.m`

Example:

`example02_filter_average.m`

Note:

This function displays a figure with the distribution of the averaged value of the frames and the gaussian fit. This figure must be manually controlled to verify that the distribution is normal, the number of outliers is below 10% and they are randomly distributed. A high number of outliers or a systematic concentration during a behavioral movement is a symptom of instability during the data acquisition.

imageRejection.m rejects the images classified as outliers and replace them by a linear interpolation of the accepted images.

Syntax:

`scanfusFilter=imageRejection(scanfus,outliers)`
`scanfusFilter=imageRejection(scanfus,outliers,method)`

Inputs:

`scanfus`, fus-structure of type fusvolume.

`outliers`, a 2D matrix of size ny by nt (planes by time) indicating the outlier frames with 1 and accepted frames with 0

`method`: is the optional interpolation method of the interp1 function of Matlab (`'linear'`, `'nearest'`, `'previous'`, `'pchirp'`), default value is `'linear'`.

Output:

`scanfusFiler`, fus-structure fo type fusvolume with the filtered data.

Example:

`example02_filter_average.m`

mapCorrelation.m computes a Pearson's correlation analysis of the functional data with a square stimulus. It allows in a minimal time to preview and control the quality of the evoked-brain activity during the experimental sessions.

Syntax:

```
correlscan=mapCorrelation(scanfus,T1,T2)
```

Inputs:

`scanfus`, fus-structure of type fusvolume or fusplane

`T1`, initial image of the stimulus,

`T2`, last image of the stimulus.

Output:

`correlscan`, fus-structure of the same type than `scanfus`. It contains the correlation maps with the stimulus. The stimulus is computed as a square window convoluted by the hemodynamic response function.

Example:

`example03_correlation.m`

registerData.m interpolates and registers a fus-structure of volume type with the Allen Mouse CCF using an affine transformation.

Syntax:

```
xreg=registerData(atlas,x,Transf)
```

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,

`x`, a fus-strucure of type volume containing the functional scan,

`Transf`, transformation structure obtained with the registering function.

Output:

`xreg`, a 3D matrix with the registered data. It has the same size orientation and voxel size than the atlas.

Example:

`example03_correlation.m`

drawBorders.m displays the edges of the brain regions from the Allen Mouse CCF for coronal, sagittal or transversal sections.

Syntax:

```
drawBorders(atlas,orientation,plane)
```

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,

`orientation` is a string containing '`coronal`', '`sagittal`' or '`transversal`' indicating the orientation of the section.

`plane`, the position of the selected plane to display.

Output:

The edges of the regions from the selected plane are displayed in the current figure.

Example:

`example03_correlation.m`

mapGlm.m computes an activity map using a generalized linear model. It uses the Matlab function `glmfit`.

Syntax:

```
[estimator,tscore]=mapGlm(X, scanfus)
```

Inputs:

`X`, matrix of regressors with size (`nt,nreg`). Each colon is a regressor of size (`nt` samples) and the model has `nreg` regressors.

`scanfus`, fus-structure of type fusvolume

Output:

`estimator`, 4D matrix of size (`nz,nx,ny,nreg`). It has the estimator of each regressor in each voxel.

`tscore`, 4D matrix of size (`nz,nx,ny,nreg`). It has the t-score of the each regressor in each voxel. Other statistics values are possible by simple modifications of the output.

Example:

`example04_glm.m`

lutSegmentation.m precomputes the interpolation, registration and segmentation of the functional scan in individual brain region.

Syntax:

```
lut=lutSegmentation(Transf,atlas,scanfus)
```

Inputs:

`Transf`, the transformation matrix, obtained with the registration function

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,

`scanfus`, fus-structure of fusvolume type,

Output:

`lut`, is a lookup table to perform a fast segmentation and registration of all the data using the same transformation. This table is used by the function `segmentation.m`.

Example:

`example05_segmentation.m`

segmentation.m performs the interpolation, registration and segmentation of the functional scan in individual brain region.

Syntax:

```
segmented=segmentation(lut,scanfus)
```

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,
`scanfus`, fus-structure of fusvolume type,
`Transf`, the transformation matrix, obtained with the registration function

Output:

`segmented`, a structure with 2 fields ('`Left`' and '`Right`') containing temporal traces for either the left or the right hemisphere. Each field is a 2D matrix of 509*nt. The 509 lines are all brain regions from the Allen Mouse CCF and nt the number of time points.

Example:

`example05_segmentation.m`

Note:

The `segmentation.m` function interpolates and registers the scan to the atlas dimension and all the voxels from the same area are added together.

selectBrainRegions.m allows to select and group brain regions to provide a readable trace map of the region activity over time.

Syntax:

```
selectSegmented=selectBrainRegions(atlas,
fileRegions,segmented)
```

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,
`fileRegions`, a sting with the name of a text file listing the selected regions, see details below,
`segmented`, structure of segmented regions provided by the `segmentation.m` function.

Output:

`selectSegmented`, a 2-field structure ('`.Left`' and '`.Right`') containing temporal traces of regions selected or grouped as organized in the `fileRegions` file for both the left and right hemisphere. Each field is a 2D matrix of size (`Nregions,nt`) where '`Nregions`' are the number of selected or grouped regions of `fileRegions` file and `nt` the time points.

Example:

`example05_segmentation`

Note:

The `fileRegions` document is a text file with a list of acronyms of the selected regions from the Allen Mouse CCF. Each line in the document is a selected region. A '//' comments the text after it. Following characters are then ignored.

This example selects 3 regions:

```
LA        // lateral amygdalar nucleus
MOp       // primary motor area
MOs       // secondary motor area
```

Lines with multiple acronyms indicate the fusion of these regions into a single region. The new region takes the acronym of the first region from the list.

This example fusions MOs into MOp:

```
MOp MOs          // region MOp is now MOp+MOs
```

A new acronym of combined regions can be given when it follows '%'. Hereafter, MO merges MOp with MOs:

```
%MO MOp MOs      // new acronym MO merges MOp and MOs
```

A readable list of brain regions is provided using the `print_region_list.m` function.

printRegionList.m generates 3 text files from the selected regions, the first ranged by alphabetical order, number of regions and volume of the region. This auxiliary function is useful to manage the list of selected regions.

Syntax:

```
printRegionList(atlas, fileRegions)
printRegionList(atlas)
```

Inputs:

`atlas`, the Allen Mouse CCF in `allen_brain_atlas.mat`,

`fileRegions`, a sting with the name of a text file listing the selected regions.

Output:

3 text files with the same names as `fileRegions`, extensions `_number.txt`, `_alpha.txt` and `_volume.txt`, containing the regions sorted alphabetically, by the number of region or the volume of the region. If the argument `fileRegions` is absent, 3 files named `atlas_alpha.txt`, `atlas_number.txt` and `atlas_volume.txt` are generated with the full list of all the atlas regions.

Example:

`example06_print_region_list.m`

Appendix 1. Stimulus example

The stimulus is triggered by a character 'T' sent by a serial port that is caught by the slave computer to generate the physical stimulus. Here are two examples on how to manage the stimulus.

Option A. Visual stimulus.

This example has been developed using the PsychoPy library that provides flexible functions to program different visual stimulus. The stimulus is generated by the slave computer. To install PsychoPy, follow the procedure 'Anaconda and Miniconda' described in `https://www.psychopy.org/download.html`.

- Copy the folder `C:\miniScan\miscellaneous\visualStim` of the master computer somewhere in the stimulus computer (i.e. `C:\visualStim`)
  Download and install Anaconda:
  `https://www.anaconda.com/products/individual.`
- Open 'Anaconda prompt' as administrator and run the following lines:
  ```
  cd C:\visualStim
  conda env create -n psychopy -f psychopy-env.yml
  ```

To control if the installation has been correctly done perform the following steps

- Open 'Anaconda prompt' and move to the folder:
  `cd C:\visualStim`
- Activate PsychoPy:
  `conda activate psychopy`
- run the test:
  `python visual_stim_test.py`

If a grating stimulus is displayed, the installation is successful.

The code consists of a loop waiting for command on a COM port. A full-screen black window is opened. When a "trigger signal" is received (default value is 'T'), the stimulus starts. When a "stop signal" (default value is '1'), the full-screen window is closed. While no stop signal is received, the window will stay open and ready to trig the stimulus. Parameters can be tuned in the code `visual_stim_utils_bar_grating.py` to generate various grating patterns. Edit the file by following the in-code documentation.

To perform a visual stimulus with the `miniScan` system, perform the following steps:

- Connect master and slave computer with a serial.
- Open the "*Device Manager*" in the master computer, select "Ports (COM & LPT)" and find the serial port. Set the port in the `miniScan` interface and open it.
- Select the same serial port in the slave computer and set it in the line 12 of `visual_stim_utils_bar_grating.py`.
- In the slave computer, open 'Anaconda prompt' and run the following lines:
  ```
  cd C:\visualStim
  conda activate psychopy
  ```

`python visual_bar_grating.py`

- Test the trigger by hitting the 'Force trigger' button in the `miniScan` interface, a vertical grid stimulus should be displayed.

Option B. TTL signal with micro-controller device.

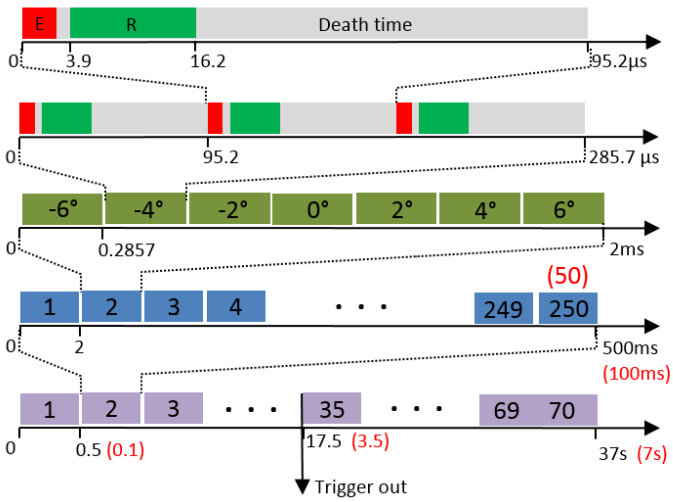
This example uses an Arduino-Uno card to receive and convert the serial port trigger into a TTL signal. This method generates a stimulus with other electronic devices through a TTL signal.

- Install the Arduino IDE
- `https://www.arduino.cc/en/Main/Software`
- Connect the Arduino card to a USB-port of the master computer and download the program '`trigArduino.io`'.
- Open the '`Device Manager`' in the master computer, check '`Ports(COM&LPT)`' and find the serial port number of the Arduino card (e.g., COM3).
- Set this port as the stimulus port in the `miniScan` software and open it.
- Test the trigger by hitting the 'Force trigger' button in the `miniScan` interface. The Arduino LED flashes and a TTL signal is sent by pin8.

Appendix 2. Ultrasound sequence

The ultrasound sequence is detailed in Figure 2:

- Emission/Reception event in which a plane wave is emitted and backscattered echoes of the brain tissue received between 3.9 and 16.25 µs after emission. This period is required to record echoes from 3 to 10mm depth. A dead time is added to reach a total time of 92.5 µs (10.5 kHz; Figure 2a).
- The E/R block is repeated 3 times to average the acquisition and thus, reduce the single to noise ratio. The total duration of a plane wave acquisition rises to 285.7 µs (3.5k Hz). The signal is beam-formed to produce a plane-wave ultrasound image (Figure 2.b).
- A set of 7 tilted plane-wave images (-6, -4, -2, 0, 2, 4, 6) generated in 2 ms (500 Hz) are coherently added to produce a high-quality compound image (Figure 2.c).
- 250 compound images (50 in fast mode) are concatenated into a single Doppler image in 0.5 s (0.1 s in fast mode; Figure 2.d).
- This set of 250 compound images is filtered to extract the signal coming from the blood. The filter consists in a 15-Hz high-pass filter and a SVD filter that eliminates 10% of the singular values. The Doppler image is then computed as the intensity of the filtered signal.
- A functional trial is composed of 70 Doppler images (35s in normal mode and 7 s in fast mode). A trigger out is sent at image 35 (17.5 s in normal mode, 3.5 s in fast mode) (Figure 2.e).



a) Emission/Reception block record echoes from 3 to 10mm of depth

b) Plane wave. 3 Signals of 3 Emission/Reception blocks are averaged to reduce SNR

c) Compound. 7 plane waves with different angles are beamformed and averaged coherently

d) µDoppler. With 250 compound (50 in fast mode) a filter selects the blood signal and the intensity is averaged

e) Trial. 70 µDoppler images are acquired to follow the variations of the blood flow.

Figure 3. Ultrasound sequence.