Beamformation Toolbox III

Generated by Doxygen 1.6.3

Wed Apr 25 15:10:53 2012

Contents

Chapter 1

Beamformation Toolbox III

Author

Jens Munk Hansen

Date

2011

Id

dummy.m,v 1.21 2011-11-08 14:03:54 jmh Exp

1.1 Table of contents

1.2 Introduction

Focusing and apodization are an essential part of signal processing in ultrasound imaging. Although the fundamental principles are simple, the dramatic increase in computational power of CPUs, GPUs, and FPGAs motivates the development of software based beamformers, which further improves image quality (and the accuracy of velocity estimation). For developing new imaging methods, it is important to establish proof-of-concept before using resources on real-time implementations. With this in mind, an effective and versatile Matlab toolbox written in C++ has been developed to assist in developing new beam formation strategies. It is a general 3D implementation capable of handling a multitude of focusing methods, interpolation schemes, and parametric and dynamic apodization. Despite being flexible, it is capable of exploiting parallelization on a single computer, on a cluster, or on both. On a single computer, it mimics the parallization in a scanner containing multiple beam formers. The focusing is determined using the positions of the transducer elements, presence of virtual sources, and the focus points. For interpolation, a number of interpolation schemes can be chosen, e.g. linear, polynomial, or cubic splines. Apodization can be specified by a number of window functions of fixed size applied on the individual elements as a function of distance to a reference point, or it can be dynamic with an expanding or contracting aperture to obtain a constant F-number, or both. On a standard PC with an Intel Quad-Core Xeon E5520 processor running at 2.26 GHz, the toolbox can beamform 300.000 points using 700.000 data samples in 3 seconds using a transducer with 192 elements, dynamic apodization in transmit and receive, and cubic splines for interpolation. This is 19 times faster than our previous toolbox.

1.3 Beamformation

Beamformation without apodization is all about delays computation for a group of signals exploiting that the sum of these signal can be either constructive or destructive. In medical ultrasound imaging, this is done for both the transmitted and the received field. The type of beamformation varies with the geometry of the transducer and the position of the focal points.

For the transmitted field, appropriate delays and possibly an apodization are applied to the transducer elements to construct a number of signals, which sum up constructively at a single focal point. Receive beamformation is similar in the sense that appropriate delays are applied to the signals received from the individual transducer elements and then a weighted sum is performed. Contrary to transmit focusing, when receiving, one can apply a number of delays corresponding to an equal number of focus points. In addition, an apodization can be applied to even out the resolution over a range of depths.

To calculate the delays, we need to compute the time-of-flight for the sound propagating from a transmit origin, to the focal points, and back to the receiving elements and convert this time to a sample index. To do the latter, we need to know the sampling frequency and the speed of sound. The two parameters are set using the bft3_system class.

```
% Setting speed-of-sound and sampling frequency
fs = 70e6;
c = 1540;
globals = bft3_system('c', c,'fs' ,fs);
```

1.4 Setting up scan-lines

An image is considered as consisting of a number of scan-lines. The scan-lines are constructed using an origin, a direction, and a length. The scan-lines are constructed using the bft3_line class.

Example

```
single_line = bft3_line([0 0 0], [0 0 1], c/fs, 40/1000);
```

1.5 Time-of-flight calculation

As just stated, we need to compute the time-of-flight for the sound propagating from the transmit origin, to the focal points and return to the receiving elements. Using the toolbox the transmit origin is set using the bft3_aperture::center_focus property on the transmit aperture, which is constructed using the constructor bft3_aperture::bft3_aperture.

Example

```
f0 = 5e6; lambda = c/f0;
pitch = lambda;
xmt_aperture = bft3_aperture('type','linear_array','pitch',pitch,...
                             'n_elements',64);
```

- Note the positions of the transmit aperture are only used for transmit apodization, see Apodization for details

1.5.1 Unfocused beams

The task of computing the time-of-flight can be split into computing a transmit and a receive time corresponding to a transmit and a receive focus, $t_{\mathrm{TOF}} = t_{\mathrm{TOF_{xmt}}} + t_{\mathrm{TOF_{rcv}}}$. Assuming the speed of sound, c is constant, we get

$$\begin{aligned} t_{\mathrm{TOF}} &= t_{\mathrm{TOF_{xmt}}} + t_{\mathrm{TOF_{rcv}}} \\ &= \frac{|\vec{r}_{\mathrm{fp}} - \vec{r}_{\mathrm{xmt}}| + |\vec{r}_{\mathrm{rcv}} - \vec{r}_{\mathrm{fp}}|}{c}. \end{aligned} \tag{1.1}$$

If secondary scattering is neglected, the receive path is a straight line and the receive time is uniquely determined. The transmit path however is not well defined, since the emitted pressure wave does not emanate from a point source $\vec{r}_{\mathrm{fp_{xmt}}}$ as indicated in Fig. 1 but rather from a complicated pattern resulting from numerous waves emitted from different elements at different times obeying Huygens' principle. For an unfocused beam though, (1) is close to correct. To consider an unfocused beam, the bft3_aperture::focus property must be empty on the transmit aperture.

```
xmt_aperture.focus = [];
```

1.5.2 Focused beams

For a focused beam, the transmit time can be approximated by considering the transmit focal point $\vec{r}_{\mathrm{fp_{xmt}}}$ as a virtual point source emitting a spherical wave. By using this approximation, the t_{TOF} becomes

$$t_{\mathrm{TOF}} = \frac{\left|\vec{r}_{\mathrm{fp_{xmt}}} - \vec{r}_{\mathrm{xmt}}\right| \pm \left|\vec{r}_{\mathrm{fp}} - \vec{r}_{\mathrm{fp_{xmt}}}\right| + |\vec{r}_{\mathrm{rcv}} - \vec{r}_{\mathrm{fp}}|}{c}, \tag{1.2}$$

where the $\pm$ in (2) refers to whether the focal point is above or below a plane orthogonal to the center line of the beam. To introduce a virtual source $\vec{r}_{\mathrm{fp_{xmt}}}$, the bft3_aperture::focus property on the transmit aperture should be set to the given position.

```
xmt_aperture.focus = [0 0 40/1000];
```

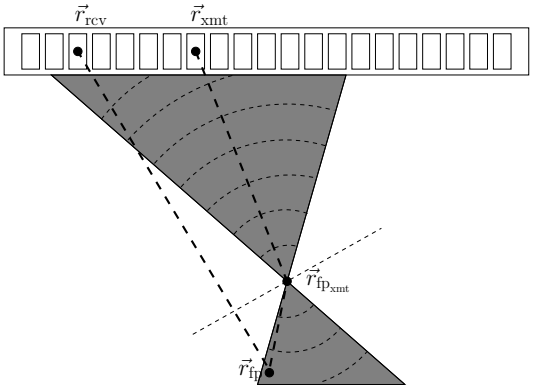


Figure 1.1: Focus point below plane

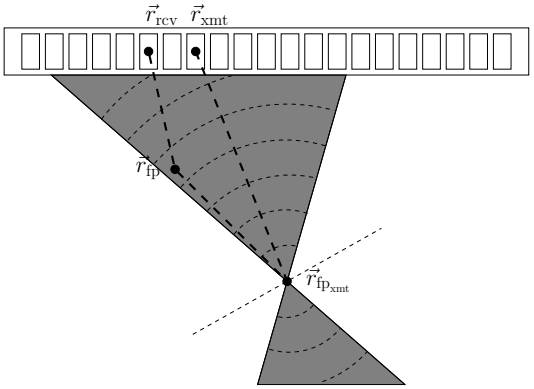


Figure 1.2: Focus point above plane

For a beam perpendicular to the aperture, the plane deciding the sign in (2) is parallel to the aperture.

1.5.3 Fixed focusing

To consider fixed receive focusing, a virtual source can be introduced for the receive aperture

```
rcv_aperture.focus = [0 0 40/1000];
```

and the t_{TOF} becomes

$$t_{\mathrm{TOF}} = \frac{\left|\vec{r}_{\mathrm{fp}_{\mathrm{xmt}}} - \vec{r}_{\mathrm{xmt}}\right| \pm \left|\vec{r}_{\mathrm{fp}} - \vec{r}_{\mathrm{fp}_{\mathrm{xmt}}}\right| + \left|\vec{r}_{\mathrm{fp}_{\mathrm{rcv}}} - \vec{r}_{\mathrm{rcv}}\right| \pm \left|\vec{r}_{\mathrm{fp}} - \vec{r}_{\mathrm{fp}_{\mathrm{rcv}}}\right|}{c}, \tag{1.3}$$

For synthetic aperture sequential beamformation (SASB), the first stage is a fixed focus beamformation stage.

1.5.4 Plane-wave focusing

For plane-wave beamformation, the $t_{\mathrm{TOF}_{\mathrm{xmt}}}$ is computed using the distance to the plane of emission and the t_{TOF} becomes

$$t_{\mathrm{TOF}} = \frac{\left|\left(\vec{r}_{\mathrm{fp}_{\mathrm{xmt}}} - \vec{r}_{\mathrm{xmt}}\right) \cdot \vec{n}\right| + \left|\vec{r}_{\mathrm{rcv}} - \vec{r}_{\mathrm{fp}}\right|}{c}, \tag{1.4}$$

In the toolbox, this plane is defined as the plane containing the transmit origin $\vec{r}_{\mathrm{xmt}}$ set using the bft3_aperture::center_focus property on the transmit aperture and perpendicular to a normal vector $\vec{n}$defined using Euler angles following the Z-X'-Z" convention (See wikipedia Euler angles). The normal vector is set using the bft3_aperture::orientation property on the transmit aperture.

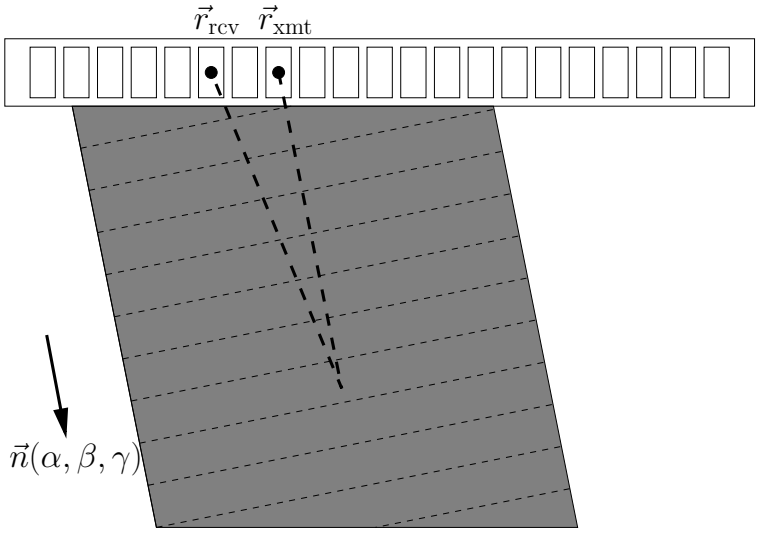


Figure 1.3: Plane-wave focusing

The result of beamforming a single point is a weighted sum of contributions for each transmit-receive channel pair. For imaging, we are interested in the absolute values and typically, we would either beamform complex data and compute the absolute or beamform a scan line of points and compute the envelope. For velocity estimation, we are interested in the phase and would therefore beamform densely sampled lines and possibly also in multiple directions, the latter for directional velocity estimation.

1.6 Apodization

In addition to focusing, beamformation also includes apodization, i.e. the possibility for tapering of the individual receive channels and also a possible overall scaling for an emission used for synthetic aperture imaging. If we include apodization, a beamformed image point at position $\vec{r}_{\mathrm{fp}}$ is computed according to

$$I\left(\vec{r}_{\mathrm{fp}}\right) = \sum_{\mathrm{xmt}=1}^{N_{\mathrm{xmt}}} \mathcal{A}_{\mathrm{xmt}}\left(\vec{r}_{\mathrm{fp}}\right) \sum_{\mathrm{rcv}=1}^{N_{\mathrm{rcv}}} \mathcal{A}_{\mathrm{rcv}}\left(\vec{r}_{\mathrm{fp}}\right) s_{\mathrm{xmt,rcv}}\left(t_{\mathrm{TOF}}\left(\vec{r}_{\mathrm{xmt}}, \vec{r}_{\mathrm{fp}_{\mathrm{xmt}}}, \vec{r}_{\mathrm{fp}}, \vec{r}_{\mathrm{rcv}}\right)\right) \tag{1.5}$$

where N_{rcv} is the number of receiving elements, $\mathcal{A}(\vec{r}_{\mathrm{fp}})$ is the apodization function in transmit and receive, and $s_{\mathrm{xmt,rcv}}(t)$ is the interpolated time-domain echo signal received at element xmt after the rcv'th emission. N_{xmt} is the number of emissions used to construct the image point, where the origin of the emissions are spatially different, which is used in synthetic transmit aperture imaging. For a conventional B-mode image, $N_{\mathrm{xmt}} = 1$.

In the toolbox the bft3_apodization class holds the defining properties of the tapering of the individual receive channels and the information for a possible overall scaling for an emission used for synthetic aperture imaging. An apodization is defined for each scan-line for both the transmitting and receiving aperture. The bft3_apodization class support two ways of tapering and the resulting apodization is the product thereof.

1.6.1 Parametric apodization

It can be defined completely in the sense that the user can define a number of windows (bft3_apodization::values) to be applied for a number of range intervals, the range being defined as the distance (bft3_apodization::distances) from an apodization reference point (bft3_apodization::ref) to the focus point. This makes it possible to a apply simple apodization or extraordinary apodization functions. Parametric apodization is enabled when the bft3_apodization::parametric property is enabled (default = true).

Example

```
% Construct apodization object to be used for a single line
rcv_apodization = bft3_apodization(rcv_aperture, [0 0 0], 0, ones(64,1));
```

1.6.2 Dynamic apodization

The second possibility is dynamic apodization with an expanding and contracting aperture defined using an F-number (bft3_apodization::f), an analytical window function (bft3_apodization::window), and the apodization reference (bft3_apodization::ref). The width of an active sub-aperture for a given focus point is then computed using the distance to the apodization reference and the F-number. If the active sub-aperture extends outside the physical aperture, then only an inner fraction of the apodization window are applied as illustrated in Fig. 3.

Apodization object are constructed using the constructor bft3_apodization::bft3_apodization by specifying an aperture, a reference point, a number of distances, and a set of apodization bft3_apodization::values. See bft3_apodization for further details.

Example

```
% Disable parametric and enable dynamic apodization
rcv_apodization.parametric = false;
rcv_apodization.dynamic = true;
rcv_apodization.f = 1.2;
```

It is important to construct two independent apodization objects for each line if multi-threading is desired. The number of execution threads is set using the bft3_image::nthreads property of the bft3_image class.

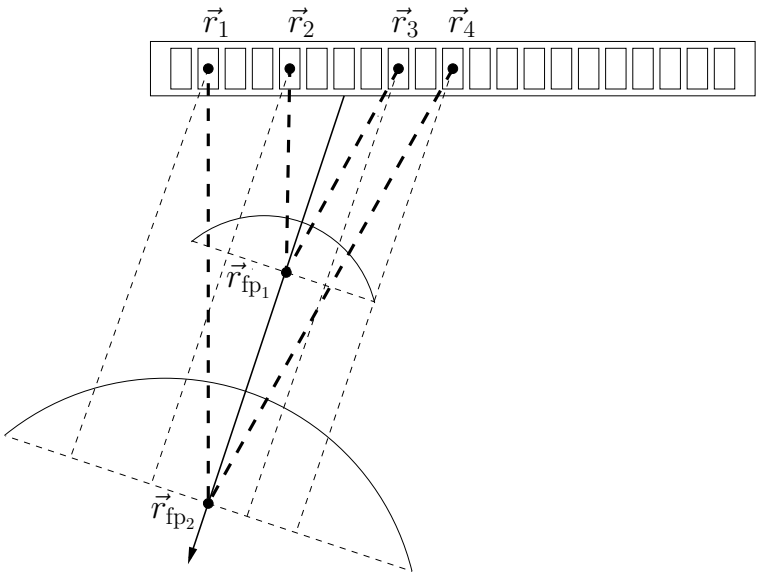


Figure 1.4: Apodization calculation

In Fig. 3, the wave propagation paths for transmit-receive element pairs for two different focal points are shown. In addition, an apodization profile is calculated corresponding to a common F-number for the two depths (The F-numbers used for transmit and receive can of course also be different as well as the window functions). The apodization values for the elements is determined by the orthogonal distance from their positions to the apodization line as indicated by the intersections between the dashed lines and the two apodization profiles. Note that for the focal point $\vec{r}_{\mathrm{fp}_2}$, we are running out of aperture and an edge-wave will most likely appear in the image. A possible way to deal with such edge-waves is to use enable both dynamic and parametric apodization and the resulting apodization will be the product.

1.6.3 Fixed width apodization

Fixed-width apodization is only available for images constructed using the bft3_sampled_image class. It is an experimental feature and only supported for apertures constructed with the bft3_aperture::type qualifier equal to 'linear_array'. The apodization is computed with a width corresponding to bft3_apodization::n_active_elements and arranged symmetrically around a line from bft3_apodization::ref to the focus point. The individual elements are then tapered according to their orthogonal distance to this line.

1.6.4 Transmit apodization

Transmit apodization can likewise be parametric or dynamic. For a dynamic apodization, the dynamic width of the sub-aperture is computed using the transmit F-number bft3_apodization::f, the distance from the focus point to the apodization reference bft3_apodization::ref. The apodization is then computed using the distance from the virtual source of the emission to the line from the apodization reference bft3_apodization::ref to the focus point. This is unfortunate for synthetic aperture beamformation using a convex array, since in this case we would like the transmit apodization to be a triangular tappering of a low-resolution image, the triangle centered around the emission. At the moment, an apodization like this can be obtained by putting all transmit apodization references equal to the beginning of the lines and adjusting the orientation property bft3_aperture::orientation of the transmit aperture in between emissions using the angle between the direction of the emission and z-axis. In this way, the apodization is calculated using the

distance from the virtual source of the emission (position of an element on the transmit aperture) to the line from the apodization reference bft3_apodization::ref to the focus point rotated according to the orientation bft3_aperture::orientation. By further scaling the transmit F-number bft3_apodization::f by a factor $1/\cos(\theta)$, where θ is the angle betwen the direction of the emission and the z-axis, the correct apodization is obtained. In the future, the transmit apodization should be changed such that it is calculated using the distance from a line defining the emission and the focus point. The reader is invited to experiment using a dataset consisting of only one value to get familiar with how transmit apodization works.

For the moment the position of the virtual sources are specified as the positions of transducer element on the transmit aperture and the position used for an emission is specified when calling the bft3_image::beamform function. The reason why we don't use the bft3_aperture::center_focus is that for an unfocused emission, we start sampling after all elements have fired and at this time the wave-front is on the surface of the aperture, whereas the virtual source of the emission is somewhere behind the transducer surface.

Example

```
% Beamformation of single-line image
img = bft3_image(xmt_aperture, rcv_aperture, xmt_apodization, rcv_apodization, s
     ingle_line);
% Dynamic transmit apodization (if enabled) uses the virtual origin located at t
     he 32'th transducer element
rf_line = img.beamform(rf_data, tstart, uint32(32));
```

1.7 Examples

A number of examples exist in the examples directory.

1.7.1 Dynamic receive focusing

An example for simulating data with Field II and beamforming an image using dynamic receive focusing is given in psf_8804p.m

1.7.2 Synthetic aperture sequential beamformation (SASB)

An example for simulating data with Field II and beamforming an image using SASB is given in psf_sasb_8804.m

1.8 A note on Matlab classes

All functions with the following prototype

function set some_class::some_property	(	in	*obj*,	
		in	*data*	
	)			

can be invoked using the assignment operator

```
some_object.some_property = data;
```

Similarly all functions with the prototype

function get some_class::some_property	(	in	*obj*	)	

can be invoked using left value assignment

```
variable_name = some_object.some_property;
```

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 bft3_aperture Class Reference

Aperture class.

Public Member Functions

- function bft3_aperture (in varargin)
  Class constructor.
- function display (in obj)
  Display function.
- function clone (in obj)
  Clone aperture.
- function get pos (in obj)
  Element positions.
- function get c (in obj)
  Speed of sound.
- function get center_focus (in obj)
  Center focus.
- function get orientation (in obj)
  Orientation.
- function get delays (in obj)
  Delays.
- function get focus (in obj)
  Focus.
- function get f0 (in obj)

Center frequency.

- function get fs (in obj)
  Sampling frequency.
- function get Id (in obj)
  Id.
- function get type (in obj)
  Type.
- function get ppwave (in obj)
  Plane-parallel wave.
- function set pos (in obj, in data)
  Set positions.
- function set center_focus (in obj, in data)
  Set center focus.
- function set orientation (in obj, in data)
  Set orientation.
- function set delays (in obj, in data)
  Set delays.
- function set focus (in obj, in data)
  Set focus.
- function set fs (in obj, in data)
  Set sampling frequency.
- function set c (in obj, in data)
  Set speed of sound.
- function set f0 (in obj, in data)
  Set center frequency.
- function set type (in obj, in data)
  Set type (read-only).
- function set ppwave (in obj, in data)
  Set ppwave option.
- function focus_delays (in obj)
  Retrieve delay values for the current focus set using the focus property.
- function delete (in obj)
  Class destructor.

- function bft3_aperture_test ()

  Unit test of the bft_aperture class.

Public Attributes

- Property mexname

  Library execution unit.

- Property Handle

  uint(32/64) object handle (read-only)

- Property Id

  uint(32/64) unique identifier of data content (read-only)

- Property type

  char string classifier (read-only)

- Property pos

  float[# elements 3] positions

- Property focus

  float[1 3] focus or virtual source

- Property center_focus

  float[1 3] reference position for time-of-flight (TOF) calculations.

- Property orientation

  float[1 3] orientation used for transmit apodization or plane-waves

- Property delays

  float[1 # elements] time delays

- Property ppwave

  bool Plane-wave option

4.1.1 Detailed Description

Aperture class. Create an Aperture object by specifying a number of options as string-arguments pairs.

Options:

'type'	char	'custom' or 'linear_array' (default: 'custom')
'n_elements'	float	# elements
'pitch'	float	pitch
'width'	float	width (only specified for 'convex_array')
'kerf'	float	kerf (only specified for 'convex_array')
'radius'	float	radius (only specified for 'convex_array')
'pos'	float	positions of elements

Properties:

pos	float[# elements, 3]	set or get positions of the elements
focus	float[1, 3]	set or get focus point (default: [])
center_focus	float[1, 3]	set or get center focus (default:center of aperture)
delays	float[1, # elements]	set or get receive delays (default: [])
ppwave	bool	enable or disable plane-wave calculation (default: false)

Read-only properties:

type	char	type of aperture
Handle	uint(32/64)	pointer
Id	uint(32/64)	unique identifier

Methods:

obj bft3_aperture (varargin)
out clone (obj)
delete (obj)
display (obj)
out focus_delays (obj), get focus delays, float[# elements, 1]

Example:

If the option 'pos' is given, e.g.

```
ah = bft3_aperture('pos',pos);
```

'n_elements', and 'pitch' are ignored.
To create a linear array facing, the xy-plane and positioned centrally and aligned with the x-axis, set the option 'type' to 'linear_array' together with the options 'pitch', and 'n_elements'.

```
fs = 30e6; f0 = 5e6; c = 1480; lambda = c/f0;
pitch = lambda;
globals = bft3_system('fs',fs,'c',c);
ah = bft3_aperture('type','linear_array','pitch',pitch,...
                   'n_elements',64);
```

To create a convex array facing the xy-plane and positioned centrally and aligned with the x-axis, set the option 'type' to 'convex_array' together with the options 'width', 'kerf', 'radius', and 'n_elements'.

```
fs = 30e6; f0 = 5e6; c = 1480; lambda = c/f0;
pitch = lambda; kerf = pitch/10; width = pitch-kerf;
radius = 20/1000;
globals = bft3_system('fs',fs,'c',c);
ah = bft3_aperture('type','convex_array','width',width,...
                   'kerf',kerf,'radius',radius,'n_elements',64);
```

Id

bft3_aperture.m,v 1.64 2011-08-30 20:05:31 jmh Exp

4.1.2 Constructor & Destructor Documentation

4.1.2.1 function bft3_aperture::bft3_aperture (in *varargin*)

Class constructor.

Create an Aperture object by specifying a number of options as string-argument pairs.

Options:

'type'	char	'custom' or 'linear_array' (default: 'custom')
'n_elements'	float	# elements
'pitch'	float	pitch
'width'	float	width (only specified for 'convex_array')
'kerf'	float	kerf (only specified for 'convex_array')
'radius'	float	radius (only specified for 'convex_array')
'pos'	float	positions of elements

Parameters

varargin a number of options as string-argument pairs

Returns

instance of the bft3_aperture class.

Example:

If the option 'pos' is given, e.g.

```
ah = bft3_aperture('pos',pos);
```

'n_elements', and 'pitch' are ignored.
To create a linear array facing, the xy-plane and positioned centrally and aligned with the x-axis, set the option 'type' to 'linear_array' together with the options 'pitch', and 'n_elements'.

```
fs = 30e6; f0 = 5e6; c = 1480; lambda = c/f0;
pitch = lambda;
globals = bft3_system('fs',fs,'c',c);
ah = bft3_aperture('type','linear_array','pitch',pitch,...
                   'n_elements',64);
```

To create a convex array facing the xy-plane and positioned centrally and aligned with the x-axis, set the option 'type' to 'convex_array' together with the options 'width', 'kerf', 'radius', and 'n_elements'.

```
fs = 30e6; f0 = 5e6; c = 1480; lambda = c/f0;
pitch = lambda; kerf = pitch/10; width = pitch-kerf;
radius = 20/1000;
globals = bft3_system('fs',fs,'c',c);
ah = bft3_aperture('type','convex_array','width',width,...
                   'kerf',kerf,'radius',radius,'n_elements',64);
```

4.1.3 Member Function Documentation

4.1.3.1 function bft3_aperture::bft3_aperture_test ()

Unit test of the bft_aperture class.

Function included for testing consistency. Throws an error in case of in-consistency

Returns

obj instance of the bft3_aperture class.

4.1.3.2 function set bft3_aperture::c (in *obj*, in *data*)

Set speed of sound.

Parameters

obj instance of the bft3_aperture class.
data float

4.1.3.3 function get bft3_aperture::c (in *obj*)

Speed of sound.

Parameters

obj instance of the bft3_aperture class.

Return values

out float speed of sound

4.1.3.4 function set bft3_aperture::center_focus (in *obj*, in *data*)

Set center focus.

Parameters

obj instance of the bft3_aperture class.
data float[1 3]

4.1.3.5 function get bft3_aperture::center_focus (in *obj*)

Center focus.

Parameters

obj instance of the bft3_aperture class.

Return values

out center focus

4.1.3.6 function bft3_aperture::clone (in *obj*)

Clone aperture.

Parameters

obj instance of the bft3_aperture class.

Return values

(deep) copy

4.1.3.7 function set bft3_aperture::delays (in *obj*, in *data*)

Set delays.

Parameters

obj instance of the bft3_aperture class.

data [1 #elements]

4.1.3.8 function get bft3_aperture::delays (in *obj*)

Delays.

Parameters

obj instance of the bft3_aperture class.

Return values

out delays

4.1.3.9 function bft3_aperture::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_aperture class.

4.1.3.10 function bft3_aperture::display (in *obj*)

Display function.

Display the properties and member functions of the class

Parameters

obj instance of the bft3_aperture class.

4.1.3.11 function set bft3_aperture::f0 (in *obj*, in *data*)

Set center frequency.

Parameters

obj instance of the bft3_aperture class.
data float

4.1.3.12 function get bft3_aperture::f0 (in *obj*)

Center frequency.

Parameters

obj instance of the bft3_aperture class.

Return values

out float center frequency

4.1.3.13 function set bft3_aperture::focus (in *obj*, in *data*)

Set focus.

Parameters

obj instance of the bft3_aperture class.
data float[1 3]

4.1.3.14 function get bft3_aperture::focus (in *obj*)

Focus.

Parameters

obj instance of the bft3_aperture class.

Return values

out float[1 3] position of focus or virtual source

4.1.3.15 function bft3_aperture::focus_delays (in *obj*)

Retrieve delay values for the current focus set using the focus property.

Parameters

obj instance of the bft3_aperture class.

Return values

focus delays corresponding to the current focus

4.1.3.16 function set bft3_aperture::fs (in *obj*, in *data*)

Set sampling frequency.

Parameters

obj instance of the bft3_aperture class.
data float

4.1.3.17 function get bft3_aperture::fs (in *obj*)

Sampling frequency.

Parameters

obj instance of the bft3_aperture class.

Return values

out float sampling frequency

4.1.3.18 function get bft3_aperture::Id (in *obj*)

Id.

Parameters

obj instance of the bft3_aperture class.

Return values

out uint(32/64) unique identifier of aperture contents

4.1.3.19 function set bft3_aperture::orientation (in *obj*, in *data*)

Set orientation.

Parameters

obj instance of the bft3_aperture class.
data float[1 3]

4.1.3.20 function get bft3_aperture::orientation (in *obj*)

Orientation.

Parameters

obj instance of the bft3_aperture class.

Return values

out orientation

4.1.3.21 function set bft3_aperture::pos (in *obj*, in *data*)

Set positions.

Parameters

obj instance of the bft3_aperture class.

data float[#elements 3]

4.1.3.22 function get bft3_aperture::pos (in *obj*)

Element positions.

Parameters

obj instance of the bft3_aperture class.

Return values

out float[#elements 3]

4.1.3.23 function set bft3_aperture::ppwave (in *obj*, in *data*)

Set ppwave option.

Parameters

obj instance of the bft3_aperture class.

data bool

4.1.3.24 function get bft3_aperture::ppwave (in *obj*)

Plane-parallel wave.

Parameters

obj instance of the bft3_aperture class.

Return values

out bool

4.1.3.25 function set bft3_aperture::type (in *obj*, in *data*)

Set type (read-only).

Parameters

obj instance of the bft3_aperture class.

data char

4.1.3.26 function get bft3_aperture::type (in *obj*)

Type.

Parameters

obj instance of the bft3_aperture class.

Return values

out char type of aperture

4.1.4 Member Data Documentation

4.1.4.1 Property bft3_aperture::center_focus

float[1 3] reference position for time-of-flight (TOF) calculations.

Set this to the position corresponding to the sample at time equal to zero. For Field II simulations, this corresponds to the position you have set using xdc_set_center_focus.

4.1.4.2 Property bft3_aperture::delays

float[1 # elements] time delays

Constant values added to any delays

4.1.4.3 Property bft3_aperture::focus

float[1 3] focus or virtual source

Set this property to a point in space float[1 3] for introducing a virtual source for the aperture. Virtual source can be introduced for the transmit or the receive aperture or both. When no virtual source is present the value should be empty

4.1.4.4 Property bft3_aperture::mexname

Library execution unit.

Name of mex-file called by the bft3_aperture class

4.1.4.5 Property bft3_aperture::ppwave

bool Plane-wave option

Enable this property for plane-wave time-of-flight (TOF) calculation. Time-of-flight is computed using the distance from the plane containing center_focus and perpendicular to bft3_aperture::orientation of the aperture to the focal points. If a virtual source is used, i.e. focus is non-zero, the time-of-flight is computed like in the case of a virtual source, but the distance from the virtual source to the focal points is now instead computed using the distance from the plane containing focus to the the focal points.

4.1.4.6 Property bft3_aperture::type

char string classifier (read-only)

String classifier set if aperture is constructed by specifying a 'type' together with a number of options. Possible apertures types include 'linear_array', 'convex_array', and 'custom'.

The documentation for this class was generated from the following file:

- bft3_aperture.m

4.2 bft3_apodization Class Reference

Apodization class.

Public Member Functions

- function bft3_apodization (in aperture, in varargin)
  Class constructor.
- function display (in obj)
  Display function.
- function clone (in obj)
  Clone apodization.
- function get ref (in obj)
  Apodization reference.
- function get distances (in obj)
  Distance to reference point (used when parametric = true).
- function get values (in obj)
  Apodization values.
- function get aperture (in obj)
  Aperture.
- function get Id (in obj)
  Unique identifier.
- function get dynamic (in obj)
  Dynamic apodization.
- function get f (in obj)
  F-number.
- function get window (in obj)
  Window.
- function get window_parameter (in obj)
  Window parameter.
- function get parametric (in obj)
  Parametric apodization.
- function set ref (in obj, in data)
  Set reference position used for parametric and dynamic apodization.
- function set distances (in obj, in data)

Set distances used for parametric apodization.

- function set values (in obj, in data)
  Set apodization windows for parametric apodization.
- function set dynamic (in obj, in data)
  Enable dynamic apodization.
- function set f (in obj, in data)
  Set F-number.
- function set parametric (in obj, in data)
  Set parametric apodization.
- function set window (in obj, in data)
  Set window (used only when dynamic = true).
- function set window_parameter (in obj, in data)
  Set window parameter (used only when dynamic = true).
- function set fixed (in obj, in data)
  Enable fixed apodization.
- function get fixed (in obj)
  Fixed apodization.
- function set n_active_elements (in obj, in data)
  Set number of active elements.
- function get n_active_elements (in obj)
  Number of active elements.
- function set orientation (in obj, in data)
  Set orientation.
- function get orientation (in obj)
  Orientation.
- function delete (in obj)
  Class destructor.
- function bft3_apodization_test ()
  Unit test of the bft_apodization class.

Public Attributes

- Property mexname
  Library execution unit.

- Property Handle

  Object handle.

- Property aperture

  Aperture object.

- Property Id

  Unique identifier of data content.

- Property ref

  Apodization reference point.

- Property parametric

  Parametric apodization enabled.

- Property distances

  Distances to reference point (used when parametric = true).

- Property values

  float[#elements, #dist] Apodization values (used when parametric = true)

- Property dynamic

  Dynamic apodization enabled.

- Property f

  F-number (used when dynamic = true).

- Property window

  Window (used when dynamic = true).

- Property window_parameter

  Window parameter (used when dynamic = true).

- Property fixed

  Enable fixed apodization (bft3_sampled image only).

- Property n_active_elements

  Number of active elements (used when fixed = true).

- Property orientation

4.2.1 Detailed Description

Apodization class. Create an apodization object using four arguments or by specifying a number of options as string-arguments pairs.

Calling:

ob = bf3_apodization(aperture, ref, distances, values)

or

ob = bft3_apodization(aperture, options)

Parameters:

aperture	class bft3_aperture	
ref is [1 3]	float	apodization reference point
distances is [1, #dist]	float	distances to reference point
values is [#elements, #dist]	float	apodization values

Options:

ref is [1,3]	float	Apodization reference point
distances is [1, #dist]	float	Distances to this reference
values is [#elements, #dist]	float	Apodization values (window functions)

ref is [1,3]	float	apodization reference point
parametric	bool	enable parametric apodization, i.e enable apodization windows each specified by a distance and a window. Each windows are active when we are further away than the distance.
distances is [1, #dist]	float	distances to reference point
values is [#elements, #dist]	float	apodization values (window functions)
fixed	bool	enable fixed apodization. Apodization window of width n_active_elements and type window with an orientation specified by 3 Euler angles (only supported for bft3_sampled_image)
n_active_elements	float	number of active elements (used when fixed = true)
dynamic	bool	enable dynamic apodization, specified by an F-number and a window type.
f	float	F-number used for dynamic apodization
window	char	window function used for dynamic and fixed apodization, 'Rectwin', 'Hamming', 'Hann', 'Blackman', 'Tukey', 'Gaussian', or 'Bartlett'
window_parameter	float	Some windows require a parameter. For the 'Gaussian' window, this is the inverse std. deviation. For the 'Tukey' window, this is ratio of taper to constant sections normalized to (0,1); 0 (Hanning), 1 (Rectwin)
orientation	float	Euler angles (not used yet)

Properties: (can be modified after construction)

Read-only properties:

aperture	bft3_aperture	associated aperture
Handle	uint(32/64)	pointer
Id	uint(32/64)	unique identifier

Methods:

obj bft3_apodization (varargin)
obj clone (obj)
delete (obj)
display (obj)

Id

bft3_apodization.m,v 1.59 2012-01-19 10:59:54 jmh Exp

4.2.2 Constructor & Destructor Documentation

4.2.2.1 function bft3_apodization::bft3_apodization (in *aperture*, in *varargin*)

Class constructor.

Create an apodization object using four arguments or by specifying a number of options as string-arguments pairs.

Calling:

ob = bf3_apodization(aperture, ref, distances, values)

or

ob = bft3_apodization(aperture, options)

Parameters:

aperture	class bft3_aperture	
ref is [1 3]	float	apodization reference point
distances is [1, # dist]	float	distances to reference point
values is [# elements, # dist]	float	apodization values

Options:

ref is [1,3]	float	Apodization reference point
distances is [1, # dist]	float	Distances to this reference
values is [# elements, # dist]	float	Apodization values (window functions)

Parameters

aperture

varargin

4.2.3 Member Function Documentation

4.2.3.1 function get bft3_apodization::aperture (in *obj*)

Aperture.

Parameters

obj instance of the bft3_apodization class

Return values

out instance of the bft3_aperture class

4.2.3.2 function bft3_apodization::bft3_apodization_test ()

Unit test of the bft_apodization class.

Function included for testing consistency

Returns

obj instance of the bft3_apodization class.

4.2.3.3 function bft3_apodization::clone (in *obj*)

Clone apodization.

Parameters

obj instance of the bft3_apodization class.

Return values

obj instance of the bft3_apodization class (deep copy)

4.2.3.4 function bft3_apodization::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_apodization class.

4.2.3.5 function bft3_apodization::display (in *obj*)

Display function.

Display the properties and member functions of the class

Parameters

obj instance of the bft3_apodization class.

4.2.3.6 function set bft3_apodization::distances (in *obj*, in *data*)

Set distances used for parametric apodization.

Parameters

obj instance of the bft3_apodization class.

data float[1, #dist]

4.2.3.7 function get bft3_apodization::distances (in *obj*)

Distance to reference point (used when parametric = true).

Parameters

obj instance of the bft3_apodization class

Return values

out float[1 # dist]

4.2.3.8 function set bft3_apodization::dynamic (in *obj*, in *data*)

Enable dynamic apodization.

Parameters

obj instance of the bft3_apodization class.
data bool

4.2.3.9 function get bft3_apodization::dynamic (in *obj*)

Dynamic apodization.

Parameters

obj instance of the bft3_apodization class

Return values

out bool

4.2.3.10 function set bft3_apodization::f (in *obj*, in *data*)

Set F-number.

Parameters

obj instance of the bft3_apodization class.
data float

4.2.3.11 function get bft3_apodization::f (in *obj*)

F-number.

Parameters

obj instance of the bft3_apodization class

Return values

out float

4.2.3.12 function get bft3_apodization::fixed (in *obj*)

Fixed apodization.

Parameters

obj instance of the bft3_apodization class.

Return values

out bool

4.2.3.13 function set bft3_apodization::fixed (in *obj*, in *data*)

Enable fixed apodization.

Parameters

obj instance of the bft3_apodization class.

data bool

4.2.3.14 function get bft3_apodization::Id (in *obj*)

Unique identifier.

Parameters

obj instance of the bft3_apodization class

Return values

out uint(32/64)

4.2.3.15 function get bft3_apodization::n_active_elements (in *obj*)

Number of active elements.

This property is only used for bft3_sampled_image and when fixed = true

Parameters

obj instance of the bft3_apodization class.

Return values

out uint32

4.2.3.16 function set bft3_apodization::n_active_elements (in *obj*, in *data*)

Set number of active elements.

This property is only used for bft3_sampled_image and when fixed = true

Parameters

obj instance of the bft3_apodization class.

data uint32

4.2.3.17 function get bft3_apodization::orientation (in *obj*)

Orientation.

Euler angles used for orientation of apodization or plane wave excitations. This option is only valid for plane-waves and/or beamformation using an bft3_sampled_image

Parameters

obj instance of the bft3_apodization class.

Return values

out float[1 3]

4.2.3.18 function set bft3_apodization::orientation (in *obj*, in *data*)

Set orientation.

Set Euler angles used for orientation of apodization. This option is not yet implemented

Parameters

obj instance of the bft3_apodization class.

data float[1 3]

4.2.3.19 function set bft3_apodization::parametric (in *obj*, in *data*)

Set parametric apodization.

Parameters

obj instance of the bft3_apodization class.

data bool

4.2.3.20 function get bft3_apodization::parametric (in *obj*)

Parametric apodization.

Parameters

obj instance of the bft3_apodization class.

Return values

out bool

4.2.3.21 function set bft3_apodization::ref (in *obj*, in *data*)

Set reference position used for parametric and dynamic apodization.

Parameters

obj instance of the bft3_apodization class.

data float[1 3]

4.2.3.22 function get bft3_apodization::ref (in *obj*)

Apodization reference.

Parameters

obj instance of the bft3_apodization class

Return values

out reference position

4.2.3.23 function set bft3_apodization::values (in *obj*, in *data*)

Set apodization windows for parametric apodization.

Parameters

obj instance of the bft3_apodization class.

data float[#elements, #dist]

4.2.3.24 function get bft3_apodization::values (in *obj*)

Apodization values.

Parameters

obj instance of the bft3_apodization class

Return values

out float[# elements # dist]

4.2.3.25 function set bft3_apodization::window (in *obj*, in *data*)

Set window (used only when dynamic = true).

Window can be either 'Rectwin', 'Hamming', 'Hann', 'Blackman', 'Tukey', 'Gaussian', or 'Bartlett'

Parameters

obj instance of the bft3_apodization class.
data char

4.2.3.26 function get bft3_apodization::window (in *obj*)

Window.

Window can be either 'Rectwin', 'Hamming', 'Hann', 'Blackman', 'Tukey', 'Gaussian', or 'Bartlett'

Parameters

obj instance of the bft3_apodization class

Return values

out char

4.2.3.27 function set bft3_apodization::window_parameter (in *obj*, in *data*)

Set window parameter (used only when dynamic = true).

Parameters

obj instance of the bft3_apodization class.
data float

4.2.3.28 function get bft3_apodization::window_parameter (in *obj*)

Window parameter.

Parameters

obj instance of the bft3_apodization class

Return values

out float

4.2.4 Member Data Documentation

4.2.4.1 Property bft3_apodization::dynamic

Dynamic apodization enabled.

This option enables a dynamic apodization using a window function specified by the window property with a width calculated using the distance to ref and the F-number f

4.2.4.2 Property bft3_apodization::fixed

Enable fixed apodization (bft3_sampled image only).

Apodization window of width n_active_elements and type window with an orientation specified by 3 Euler angles on the respective aperture

4.2.4.3 Property bft3_apodization::mexname

Library execution unit.

Name of mex-file called by the bft3_apodization class

4.2.4.4 Property bft3_apodization::n_active_elements

Number of active elements (used when fixed = true).

Width of apodization window (when fixed = true), the pitch is taken as the distance between the first two elements on the receive aperture, hence it only works for bft3_aperture's constructed as 'linear array"s

4.2.4.5 Property bft3_apodization::orientation

Orientation (used when fixed = true or when the bft3_aperture::ppwave property is enabled on the aperture)

4.2.4.6 Property bft3_apodization::parametric

Parametric apodization enabled.

Enable apodization windows each specified by a distance in distances and a window in values. A window is active when we are further away from ref than the distance.

4.2.4.7 Property bft3_apodization::ref

Apodization reference point.

This point is used as a reference for distances used for parametric apodization and together with the f number for the calculation of the size of an active sub-aperture for dynamic apodization.

4.2.4.8 Property bft3_apodization::values

float[#elements, #dist] Apodization values (used when parametric = true)

A set of values is specified for each distance used for parametric apodization.

4.2.4.9 Property bft3_apodization::window

Window (used when dynamic = true).

Window function used for dynamic and fixed apodization. Valid windows are: 'Rectwin', 'Hamming', 'Hann', 'Blackman', 'Tukey', 'Gaussian', or 'Bartlett'

4.2.4.10 Property bft3_apodization::window_parameter

Window parameter (used when dynamic = true).

Some windows require a parameter. For the 'Gaussian' window, this is the inverse std. deviation. For the 'Tukey' window, this is ratio of taper to constant sections normalized to (0,1); 0 (Hanning), 1 (Rectwin)

The documentation for this class was generated from the following file:

- bft3_apodization.m

4.3 bft3_im_geom Class Reference

Image Geometry Class.

Public Member Functions

- function bft3_im_geom (in varargin)
  Class constructor.
- function display (in obj)
  Display function.
- function x (in obj, in varargin)
  x coordinates
- function y (in obj, in varargin)
  y coordinates
- function z (in obj, in varargin)
  z coordinates
- function np (in obj)
  Number of pixels.
- function circ (in obj, in rad, in over)
  Circle for masking.
- function bft3_im_geom_test ()
  Unit test of the bft_im_geom class.

Public Attributes

- Property nx
  Number of x-pixels.
- Property ny
  Number of y-pixels.
- Property nz
  Number of z-pixels.
- Property dx
  Pixel separation, x-direction.
- Property dy
  Pixel separation, y-direction.
- Property dz

Pixel separation, z-direction.

- Property offset_x

  Pixel offset in units of dx.

- Property offset_y

  Pixel offset in units of dy.

- Property offset_z

  Pixel offset in units of dz.

- Property fov

  Field of view (FOV).

- Property mask

  Image mask.

- Property dim

  Dimensions.

4.3.1 Detailed Description

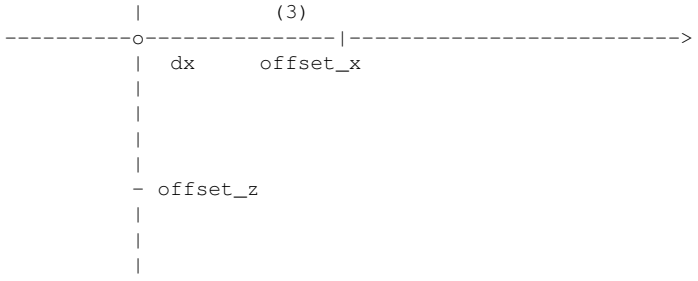
Image Geometry Class. Create a "image geometry" class that describes the sampling characteristics of a single 2D image.

Options:

'nx'	float	image dimension
'nz'	float	image dimension (default: nx)
'dx'	float	pixel size (required)
'dz'	float	pixel size (default: -dx)
'offset_x'	float	[units of dx] (default: 0)
'offset_z'	float	[units of dz] (default: 0)
'fov'	float	nx ∗ dx
'mask'	float	logical support mask

Methods:

x	float[1 np]	1D x coordinates of each pixel
y	float[1 np]	1D y
z	float[1 np]	1D z
np	float	sum(mask(:)) (# of pixels to be estimated)
circ	float[nx nz]	2D image with ellipsis



```
              v

dx and dz specifies direction of axes
```

Id

bft3_im_geom.m,v 1.13 2011-08-04 18:18:04 jmh Exp

4.3.2 Constructor & Destructor Documentation

4.3.2.1 function bft3_im_geom::bft3_im_geom (in *varargin*)

Class constructor.

Create an "image geometry" object that describes the sampling characteristics of a single 2D image. The object is string-arguments pairs.

Options:

'nx'	float	image dimension
'nz'	float	image dimension (default: nx)
'dx'	float	pixel size (required)
'dz'	float	pixel size (default: -dx)
'offset_x'	float	[units of dx] (default: 0)
'offset_z'	float	[units of dz] (default: 0)
'fov'	float	nx ∗ dx
'mask'	float	logical support mask

varargin

4.3.3 Member Function Documentation

4.3.3.1 function bft3_im_geom::bft3_im_geom_test ()

Unit test of the bft_im_geom class.

Function included for testing consistency

Returns

obj instance of the bft3_im_geom class.

4.3.3.2 function bft3_im_geom::circ (in *obj*, in *rad*, in *over*)

Circle for masking.

Parameters

obj instance of the bft3_im_geom class

4.3.3.3 function bft3_im_geom::display (in *obj*)

Display function.

Parameters

obj instance of the bft3_im_geom class

4.3.3.4 function bft3_im_geom::np (in *obj*)

Number of pixels.

Parameters

obj instance of the bft3_im_geom class

4.3.3.5 function bft3_im_geom::x (in *obj*, in *varargin*)

x coordinates

Parameters

obj instance of the bft3_im_geom class

varargin indices (if any)

4.3.3.6 function bft3_im_geom::y (in *obj*, in *varargin*)

y coordinates

Parameters

obj instance of the bft3_im_geom class

varargin indices (if any)

4.3.3.7 function bft3_im_geom::z (in *obj*, in *varargin*)

z coordinates

Parameters

obj instance of the bft3_im_geom class

varargin indices (if any)

The documentation for this class was generated from the following file:

- bft3_im_geom.m

4.4 bft3_image Class Reference

Image class.

Public Member Functions

- function bft3_image (in xmt_aperture, in rcv_aperture, in xmt_apodizations, in rcv_apodizations, in bft_lines, in varargin)

  Class constructor.

- function beamform (in obj, in rf_data, in delays, in i_xmt)

  Beamform.

- function set interp (in obj, in data)

  Set interpolation type.

- function get interp (in obj)

  Interpolation type.

- function set nthreads (in obj, in data)

  Set number of execution threads.

- function get nthreads (in obj)

  Number of execution threads.

- function display (in obj)

  Display function.

- function delete (in obj)

  Class destructor.

- function bft3_image_test ()

  Unit test of the bft3_image class.

Public Attributes

- Property mexname

  Library execution unit.

- Property Handle

  Handle.

- Property interp

  Interpolation type.

- Property nthreads

  Number of execution threads.

4.4.1 Detailed Description

Image class. ob = bft3_image(aperture, aperture, apodizations, apodizations, lines, options)

Create an image object using two apertures, two arrays of apodizations, an array of lines and a number of options as string-arguments pairs.

Options:

Properties: (can be modified after construction)

interp	char	'nearest'	nearest neighbour interpolation
		'linear'	linear interpolation
		'cubic'	the four nearest points are used for fitting a cubic polynomial
		'spline'	natural cubic splines
		'fir'	upsampling by a factor of 8, using a predefined LP FIR filter of order=48
nthreads	uint32		number of execution threads or beamformers

Read-only properties:

Handle	uint(32/64)	pointer

Methods:

beamform (rf_data, delay, i_xmt)

rf_data	float[# elements # samples]	
delay	float	
i_xmt	uint32	Transmission number i_xmt is only used for dynamic transmit apodization.

Id

bft3_image.m,v 1.32 2011-07-25 15:55:14 jmh Exp

4.4.2 Constructor & Destructor Documentation

4.4.2.1 function bft3_image::bft3_image (in *xmt_aperture*, in *rcv_aperture*, in *xmt_apodizations*, in *rcv_apodizations*, in *bft_lines*, in *varargin*)

Class constructor.

Parameters

xmt_aperture instance of bft3_aperture class

rcv_aperture instance of bft3_aperture class

xmt_apodizations bft3_apodization[# lines]
rcv_apodizations bft3_apodization[# lines]
bft_lines bft3_lines[# lines]
varargin

Returns

instance of the bft3_image class.

4.4.3 Member Function Documentation

4.4.3.1 function bft3_image::beamform (in *obj*, in *rf_data*, in *delays*, in *i_xmt*)

Beamform.

Parameters

obj instance of the bft3_image class.
rf_data float[# rf_samples # channels]
delays float time of first sample
i_xmt uint32 index specifying origin of emission (used for dynamic transmit apodization only)

Return values

out float[# lines # samples]

4.4.3.2 function bft3_image::bft3_image_test ()

Unit test of the bft3_image class.

Function included for testing consistency. Throws an error in case of in-consistency

Returns

obj instance of the bft3_image class

4.4.3.3 function bft3_image::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_image class.

4.4.3.4 function bft3_image::display (in *obj*)

Display function.

Parameters

obj instance of the bft3_image class.

4.4.3.5 function get bft3_image::interp (in *obj*)

Interpolation type.

Return interpolation used

Parameters

obj instance of the bft3_image class

Return values

out char

4.4.3.6 function set bft3_image::interp (in *obj*, in *data*)

Set interpolation type.

data can be either 'nearest', 'linear', 'cubic', 'spline', or 'fir

Parameters

obj instance of the bft3_image class.

data char

4.4.3.7 function get bft3_image::nthreads (in *obj*)

Number of execution threads.

Parameters

obj instance of the bft3_image class

Return values

out

4.4.3.8 function set bft3_image::nthreads (in *obj*, in *data*)

Set number of execution threads.

Parameters

obj instance of the bft3_image class.

data input

4.4.4 Member Data Documentation

4.4.4.1 Property bft3_image::interp

Interpolation type.

Interpolation can be done using either 'nearest', 'linear', 'cubic', 'spline', or 'fir interpolation

4.4.4.2 Property bft3_image::mexname

Library execution unit.

Name of mex-file called by the bft3_image class

4.4.4.3 Property bft3_image::nthreads

Number of execution threads.

Number of threads, the scheduler is starting the first threads in a cyclic order on the cores available on your system. If any cores are hyperthreaded, they are selected as the last cores used before multiple threads are executed on any core.

The documentation for this class was generated from the following file:

- bft3_image.m

4.5 bft3_line Class Reference

Line class.

Public Member Functions

- function bft3_line (in varargin)
  Class constructor.
- function get **origin** (in obj)
- function get **direction** (in obj)
- function get **dr** (in obj)
- function get xmt_apodization (in obj)
  Transmit apodization.
- function get rcv_apodization (in obj)
  Receive apodization.
- function pos (in obj, in varargin)
  Points on the line.
- function xmt_apodization_values (in obj, in index)
  Transmit apodization values.
- function rcv_apodization_values (in obj)
  Receive apodization values.
- function display (in obj)
  Display function.
- function delete (in obj)
  Class destructor.
- function bft3_line_test ()
  Unit test of the bft_line class.

Public Attributes

- Property mexname
  Library execution unit.
- Property Handle
  Handle.
- Property origin
  Origin of line.
- Property direction

Direction unit vector.

- Property dr

  Line increment.

- Property xmt_apodization

  Transmit apodization (read-only).

- Property rcv_apodization

  Receive apodization (read-only).

4.5.1 Detailed Description

Line class. Line class for Beam Formation

Calling

ob = bft3_line(origin, direction, dr, length)

Parameters:

origin	float[1,3]	Starting point of line
direction	float[1,3]	Unit vector
dr	float	Size of increment
length	float	Length

Methods:

obj bft3_line(varargin)
delete(obj)
display(obj)
out pos(obj, varargin)

Id

bft3_line.m,v 1.37 2012-03-26 09:34:07 jmh Exp

4.5.2 Constructor & Destructor Documentation

4.5.2.1 function bft3_line::bft3_line (in *varargin*)

Class constructor.

Calling

ob = bft3_line(origin, direction, dr, length)

Parameters:

origin	float[1,3]
direction	float[1,3]
dr	float
length	float

Parameters

varargin 4 parameters

Returns

instance of the bft3_line class.

4.5.3 Member Function Documentation

4.5.3.1 function bft3_line::bft3_line_test ()

Unit test of the bft_line class.

Function included for testing consistency. Throws an error in case of in-consistency

Returns

obj instance of the bft3_line class.

4.5.3.2 function bft3_line::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_image class.

4.5.3.3 function bft3_line::display (in *obj*)

Display function.

Display the properties and member functions of the class

Parameters

obj instance of the bft3_line class

4.5.3.4 function bft3_line::pos (in *obj*, in *varargin*)

Points on the line.

Parameters

obj instance of the bft3_line class.

varargin optional indices, size of point are [#samples 3]

Return values

out float[]

4.5.3.5 function get bft3_line::rcv_apodization (in *obj*)

Receive apodization.

Parameters

obj instance of the bft3_line class.

Return values

out instance of the bft3_apodization class

4.5.3.6 function bft3_line::rcv_apodization_values (in *obj*)

Receive apodization values.

Retrieve receive apodization values

Parameters

obj instance of the bft3_line class.

Return values

out float[#rcv_channels #samples]

4.5.3.7 function get bft3_line::xmt_apodization (in *obj*)

Transmit apodization.

Parameters

obj instance of the bft3_line class.

Return values

out instance of the bft3_apodization class

4.5.3.8 function bft3_line::xmt_apodization_values (in *obj*, in *index*)

Transmit apodization values.

Retrieve transmit apodization values for the emission originating from position index of the transmit aperture

Parameters

obj instance of the bft3_line class.

index of transmit origin

Return values

out float[#samples]

4.5.4 Member Data Documentation

4.5.4.1 Property bft3_line::mexname

Library execution unit.

Name of mex-file called by the bft3_line class

4.5.4.2 Property bft3_line::rcv_apodization

Receive apodization (read-only).

(debug purposes only)

4.5.4.3 Property bft3_line::xmt_apodization

Transmit apodization (read-only).

(debug purposes only)

The documentation for this class was generated from the following file:

- bft3_line.m

4.6 bft3_sampled_image Class Reference

Sampled Image class.

Public Member Functions

- function bft3_sampled_image (in varargin)
  Class constructor.
- function beamform (in obj, in rf_data, in delay, in)
  Beamform.
- function set interp (in obj, in data)
  Set interpolation type.
- function get interp (in obj)
  Interpolation type.
- function set nthreads (in obj, in data)
  Set number of execution threads.
- function get nthreads (in obj)
  Get number of execution threads.
- function display (in obj)
  Display function.
- function delete (in obj)
  Class destructor.
- function bft3_sampled_image_test ()
  Test of bft3_sampled_image class.

Public Attributes

- Property mexname
  Library execution unit.
- Property Handle
  Handle.
- Property interp
  Interpolation type.
- Property nthreads
  Number of execution threads.

4.6.1 Detailed Description

Sampled Image class. ob = bft3_sampled_image(aperture, aperture, im_geom)

Create an image object using two apertures, an im_geom object, and a number of options as string-arguments pairs.

Options:

Properties: (can be modified after construction) interp Options are 'nearest' - nearest neighbour interpolation 'linear' - linear interpolation 'cubic' - the four nearest points are used for fitting a cubic polynomial 'spline' - natural cubic splines 'fir' - upsampling by a factor of 8, using a predefined LP FIR filter of order=48

nthreads uint32, number of threads or beamformers

Protected properties: Handle Id

Methods:

Id

bft3_sampled_image.m,v 1.20 2011-04-27 20:35:28 jmh Exp

4.6.2 Constructor & Destructor Documentation

4.6.2.1 function bft3_sampled_image::bft3_sampled_image (in *varargin*)

Class constructor.

Parameters

varargin a number of options as string-argument pairs

Returns

instance of the bft3_sampled_image class.

4.6.3 Member Function Documentation

4.6.3.1 function bft3_sampled_image::beamform (in *obj*, in *rf_data*, in *delay*, in)

Beamform.

Parameters

obj instance of the bft3_sampled_image class.

rf_data

delay

i_xmt

angles

Return values

4.6.3.2 function bft3_sampled_image::bft3_sampled_image_test ()

Test of bft3_sampled_image class.

Function included for testing consistency

Returns

obj instance of the bft3_sampled_image class

4.6.3.3 function bft3_sampled_image::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_sampled_image class.

4.6.3.4 function bft3_sampled_image::display (in *obj*)

Display function.

Parameters

obj instance of the bft3_sampled_image class.

4.6.3.5 function get bft3_sampled_image::interp (in *obj*)

Interpolation type.

Return interpolation used

Parameters

obj instance of the bft3_image class

Return values

out char

4.6.3.6 function set bft3_sampled_image::interp (in *obj*, in *data*)

Set interpolation type.

Parameters

obj instance of the bft3_sampled_image class.

data input

4.6.3.7 function get bft3_sampled_image::nthreads (in *obj*)

Get number of execution threads.

Parameters

obj instance of the bft3_sampled_image class

Return values

out

4.6.3.8 function set bft3_sampled_image::nthreads (in *obj*, in *data*)

Set number of execution threads.

Parameters

obj instance of the bft3_sampled_image class.

data input

4.6.4 Member Data Documentation

4.6.4.1 Property bft3_sampled_image::interp

Interpolation type.

Interpolation can be done using either 'nearest', 'linear', 'cubic', 'spline', or 'fir interpolation

4.6.4.2 Property bft3_sampled_image::mexname

Library execution unit.

Name of mex-file called by the bft3_sampled_image class

4.6.4.3 Property bft3_sampled_image::nthreads

Number of execution threads.

Number of threads, the scheduler is starting the first threads in a cyclic order on the cores available on your system. If any cores are hyperthreaded, they are selected as the last cores used before multiple threads are executed on any core.

The documentation for this class was generated from the following file:

- bft3_sampled_image.m

4.7 bft3_system Class Reference

System class.

Public Member Functions

- function bft3_system (in varargin)
  Class constructor.
- function display (in obj)
  Display function.
- function get fs (in obj)
  Sampling frequency.
- function get c (in obj)
  Get the speed of sound.
- function get version (in obj)
  Version.
- function set fs (in obj, in data)
  Set sampling frequency.
- function set c (in obj, in data)
  Set speed of sound.
- function delete (in obj)
  Class destructor.
- function bft3_system_test ()
  Unit test of the bft3_system class.

Public Attributes

- Property mexname
  Library execution unit.
- Property fs
  float Sampling frequency
- Property c
  float Speed of sound
- Property version
  char package version

4.7.1 Detailed Description

System class. See class description Create a system object containing information about sampling frequency fs and the speed of sound c. The object is created by specifying a number of options as string-argument pairs. This object must be created for any beamformation scenario

ob = bft3_system(options)

Options:

'fs' float, sampling rate
'c' float, speed of sound

Properties:

'fs' float, set or get sampling rate
'c' float, set or get speed of sound

Example:

```
fs = 30e6; c = 1480;\n
globals = bft3_system('fs',fs,'c',c);
```

Id

bft3_system.m,v 1.25 2011-08-02 18:53:51 jmh Exp

4.7.2 Constructor & Destructor Documentation

4.7.2.1 function bft3_system::bft3_system (in *varargin*)

Class constructor.

Create a system object by specifying a number of options as string-argument pairs.

ob = bft3_system(options)

Options:

'fs' float, sampling rate
'c' float, speed of sound

Properties:

'fs' float, set or get sampling rate
'c' float, set or get speed of sound

Example:

fs = 30e6; c = 1480;
globals = bft3_system('fs',fs,'c',c);

Parameters

varargin a number of options as string-argument pairs

Return values

obj instance of the bft3_system class.

4.7.3 Member Function Documentation

4.7.3.1 function bft3_system::bft3_system_test ()

Unit test of the bft3_system class.

Function included for testing consistency. Throws an error in case of in-consistency

Returns

obj instance of the bft3_system class.

4.7.3.2 function set bft3_system::c (in *obj*, in *data*)

Set speed of sound.

Parameters

obj instance of the bft3_system class.

data float

4.7.3.3 function get bft3_system::c (in *obj*)

Get the speed of sound.

Parameters

obj instance of the bft3_system class.

Return values

out float speed of sound

4.7.3.4 function bft3_system::delete (in *obj*)

Class destructor.

Delete method are called before an object of the class is destroyed

Parameters

obj instance of the bft3_system class.

4.7.3.5 function bft3_system::display (in *obj*)

Display function.

Display the properties and member functions of the class

Parameters

obj instance of the bft3_system class.

4.7.3.6 function set bft3_system::fs (in *obj*, in *data*)

Set sampling frequency.

Parameters

obj instance of the bft3_system class.

data float

4.7.3.7 function get bft3_system::fs (in *obj*)

Sampling frequency.

Parameters

obj instance of the bft3_system class.

Return values

out float sampling frequency

4.7.3.8 function get bft3_system::version (in *obj*)

Version.

Get the version string for the toolbox matching the CVS tag used for compilation

Parameters

obj instance of the bft3_system class.

Return values

out char version string

The documentation for this class was generated from the following file:

- bft3_system.m

Chapter 5

File Documentation

5.1 bft3_apodizations.m File Reference

Construct multiple apodization objects in one go.

Functions

- function bft3_apodizations (in aperture, in varargin)
  Function for constructing multiple apodizations in one go.
- function bft3_apodizations_do (in aperture, in ref, in distances, in values, in st)
  Internal function.
- function **bft3_apodizations_test** ()

5.1.1 Detailed Description

Construct multiple apodization objects in one go.

5.1.2 Function Documentation

5.1.2.1 function bft3_apodizations (in *aperture*, in *varargin*)

Function for constructing multiple apodizations in one go.

Function for constructing multiple apodizations in one go using the aperture given as the first argument and a number of options given as string-argument pairs, where the strings equal any of the following: ref, distances, values, dynamic, parametric, fixed, window, window_parameter, f, n_active_elements - similar to when constructing single apodization objects.

Returns

array of instances of the bft3_apodization class.

5.2 bft3_caller_name.m File Reference

Return name and line of calling routine.

Functions

- function bft3_caller_name (in level)

5.2.1 Detailed Description

Return name and line of calling routine.

5.2.2 Function Documentation

5.2.2.1 function bft3_caller_name (in *level*)

function [name, line] = bft3_caller_name(level)

return name (and line) of calling routine or file (if level=1, the default) or name further up or down the stack by changing caller

5.3 bft3_lines.m File Reference

Construct multiple line objects in one go.

Functions

- function bft3_lines (in varargin)
  Function for constructing multiple lines.
- function bft3_lines_viewport_do (in viewport)
  Internal function.
- function bft3_lines_do (in origins, in directions, in drs, in lengths)
  Internal function.
- function bft3_lines_test ()
  Internal function.

5.3.1 Detailed Description

Construct multiple line objects in one go.

5.3.2 Function Documentation

5.3.2.1 function bft3_lines (in *varargin*)

Function for constructing multiple lines.

If any values are missing, they are duplicated, e.g. if only one origin is given and multiply directions, lines will be constructed with the same origin and multiple directions

Calling:

ob = bft3_line(origin, direction, dr, length)

or

ob = bf3_apodization(options)

Parameters:

origins	float[#lines,3]
direction	float[#lines,3]
dr	float
length	float[#lines]

Id

bft3_lines.m,v 1.14 2011-08-02 18:53:51 jmh Exp

Index