Appendix B

CM/AVS Modules

This appendix contains man pages for the following CM/AVS modules, in alphabetical order:

`antialias cm`	**`fft cm`**
`clamp cm`	**`field math cm`**
`color range cm`	**`field to byte cm`**
`colorizer cm`	**`field to double cm`**
`combine scalars cm`	**`field to float cm`**
`compare field cm`	**`field to int cm`**
`compute gradient cm`	**`luminance cm`**
`contrast cm`	**`orthogonal slicer cm`**
`downsize cm`	**`threshold cm`**
`extract scalar cm`	

With the exception of **`fft cm`**, all the CM/AVS modules are AVS modules that have been adapted for the CM-5, and they may be interchanged with their AVS counterparts. For example, **`downsize cm`** and **`downsize`** are interchangeable.

See Appendix C and **`/usr/examples/cmavs/unsupported/README`** for information about unsupported modules.

B.1 antialias cm

Antialias an image.

Type	filter
Inputs	field 2D uniform 4-vector byte (*image*)
Outputs	field 2D uniform 4-vector byte (*image*)
Parameters	none

The **antialias cm** module downsamples an image using a Gaussian 3x3 convolution filter. This produces an antialiasing effect, reducing jagged edges. The output image is half the size of the input image in each dimension.

B.2 clamp cm

Restrict values in data field to user-specified range.

Type	filter
Inputs	field *any-dimension n-vector any-data any-coordinates*
Outputs	field of same type as input

Parameters	**Name**	**Type**	**Default**	**Min**	**Max**
	clamp_min	float	0.0	none	none
	clamp_max	float	255.0	none	none

The **clamp cm** module transforms the values of a field as follows:

- Any value less than the value of the **clamp_min** parameter is set to **clamp_min**.
- Any value greater than the value of the **clamp_max** parameter is set to **clamp_max**.
- Values within the **clamp_min**-to-**clamp_max** range are not changed.

After being **clamp**'ed, a data set's values are all in this range:

clamp_min <= *value* <= **clamp_max**

B.3 color range cm

Scale AVS colormap to the range of data in a parallel field.

Type data

Inputs field (*any-dimension* scalar *any-data any-coordinates*)
colormap

Outputs colormap

Parameters none

The **color range cm** module adjusts the minimum and maximum values of a colormap to those of a parallel field, thus normalizing the colormap to the range of the data in the field.

Use **color range cm** whenever you have data that you want represented as colors, but that data's range of values is either not evenly distributed between 0 and 255, or many of the data values lie outside the 0 to 255 range.

B.4 colorizer cm

Convert field of data values to color values.

Type filter

Inputs field *any-dimension* scalar *any-data any-coordinates* colormap

Outputs field *any-dimension* 4-vector byte *any-coordinates*

Parameters none

The **colorizer cm** module converts the data at each point of a scalar field from the input value (which can be any data type) to a *color* (4-vector of bytes). The conversion is accomplished by using the input value as an index into a colormap.

B.5 combine scalars cm

Combine scalar fields into a vector field.

Type filter

Inputs field *any-dimension* scalar *any-data any-coordinates* (channel 0 — optional)

field *any-dimension* scalar *any-data any-coordinates* (channel 1 — optional)

field *any-dimension* scalar *any-data any-coordinates* (channel 2 — optional)

field *any-dimension* scalar *any-data any-coordinates* (channel 3 — optional)

Outputs field *same-dimension* 1D−4D *same-data*

Parameters

Name	Type	Default	Min	Max
Vector	Len	Dial	4	14

The **combine scalars cm** module combines up to four fields with scalar data values into a field whose data values are vectors. The input field must be of like dimension, and the scalar values must be of the same type.

This module is generally most useful for constructing images or gradient fields from separately computed components.

B.6 compare field cm

Compare two fields, display and write data difference.

Type data output

Inputs field *any-dimension n-vector any-data any-coordinates*
field *same-dimension same-vector same-data same-coordinates*

Outputs none

Parameters	**Name**	**Type**	**Default**	**Min**	**Max**
	Do Compare	oneshot	off		
	Max Elements	integer	100	1	1000
	Output File	typein	*/tmp/cfield_...*		

The **compare field cm** module compares any two identically-structured AVS fields. It prints out differences between the headers if they are different. If the headers are the same, it proceeds to do a comparison of the data contents of the two fields. If the fields are not identical in their data components, **compare field cm** prints the message, "fields are DIFFERENT," to standard output.

The **compare field cm** module was originally written to make sure that two identical modules, one written in C and one written in Fortran, produced the same results. It could also be useful to compare the contents of a field before and after an operation has been performed on it.

B.7 compute gradient cm

Compute gradient vectors for 2D or 3D data set.

Type filter

Inputs field 2D/3D scalar byte *any-coordinates*

Outputs field *same-dimension* 3-vector real *same-coordinates*

Parameters	**Name**	**Type**	**Default**	**Min**	**Max**
	2D Height	float	0.5	0.0	1.0
	Flip	toggle	on	off	on

The **compute gradient cm** module computes the gradient vector at each point in a 2D or 3D field of data. The gradient can be used (e.g. by **gradient shade**) as a "pseudo surface normal" at each point.

B.8 contrast cm

Perform linear transformation on range of field values.

Type	filter
Inputs	field *any-dimension n-vector any-data any-coordinates*
Outputs	field of same type as input

Parameters	Name	Type	Default	Min	Max
	cont_in_min	float	0.0	none	none
	cont_in_max	float	255.0	none	none
	cont_out_min	float	0.0	none	none
	cont_out_max	float	255.0	none	none

The **contrast cm** module transforms all the values in a field. Two different types of transformation take place:

- All values that fall within the "input range" specified by the **cont_in_min** and **cont_in_max** parameters are transformed linearly to the "output range" specified by **cont_out_min** and **cont_out_max**. In essence, this transformation "stretches" or "compresses" one specified range of data to fit another specified range.
- All values that fall outside the specified input range are "clamped" to the limit values of the output range.

The **contrast cm** module typically is used to remove low-level noise from images and volumes, or to increase the contrast in faded images and volumes.

B.9 downsize cm

Reduce size of data set by sampling.

Type	filter
Inputs	field 2D/3D *n-vector any-data any-coordinates*
Outputs	field of same type as input

Parameters	Name	Type	Default	Min	Max
	downsize	integer	8	1	16

The **downsize cm** module changes the size of the input data set by subsampling the data. It extracts every *n*th element of the field along each dimension, where *n* is the value of the **downsize factor** parameter. This technique preserves the aspect ratio of the input data.

This module is useful for operating on a reduced amount of data, in order to adjust other processing parameters interactively, or save memory.

B.10 extract scalar cm

Extract a scalar field from a vector field.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates* (*n* = 1..25)

Outputs field *same-dimension* scalar *same-data same-coordinates*

Parameters	Name	Type	Default
	Channel *n*	radio buttons	Channel 0

The **extract scalar cm** module inputs a field whose data values are vectors (1D to 25D), and outputs one of the dimensions ("channels") as a scalar-valued field. The output field has the same structure as the input field, except that its data values are scalars (vector length of 1).

This module is useful for performing operations on individual channels of vector fields.

B.11 fft cm

Do a Fast Fourier Transform on a field.

Type filter

Inputs field float (each axis must be length power of two)

Outputs field

Parameters op

The **fft cm** module takes a floating point {1|2|3}D 2-vector and, depending on the "op" parameter, does either a forward or inverse Fast Fourier Transform on it. The module uses the simple FFT routine in the Thinking Machines CMSSL library.

B.12 field math cm

Perform math operations between fields.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates*
field *same-dimension same-vector any-data same-coordinates* (OPTIONAL)

Outputs field *same-dimension same-vector any-data same-coordinates*

Parameters

Name	Type	Default	Min	Max
choice	choice	+		
Normalize	boolean	off		
Constant	float typein	0.0	unbounded	unbounded

The **field math cm** module performs unary and binary operations upon parallel fields.

The unary operations are Not, Square, and Sqrt. The binary operations are +, –, *, /, And, Or, Xor, Left-Shift, Right-Shift, and RMS (Root Mean Square).

B.13 field to byte cm

Transform any field to a byte-valued field.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates*

Outputs field *same-dimension same-vector* byte *any-coordinates*

Parameters	**Name**	**Type**	**Default**	**Choices**
	byte normalize	toggle	on	on, off

The **field to byte cm** module takes a field of data (*integer*, *real*, *double*, or *byte*) and converts it to a *byte* field. It can be used in conjunction with volume visualization modules that have a bias towards *byte* fields.

B.14 field to double cm

Transform any field to a field of double-precision floating point values.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates*

Outputs field *same-dimension same-vector* double *same-coordinates*

Parameters	**Name**	**Type**	**Default**	**Choices**
	double normalize	toggle	on	on, off

The **field to double cm** module takes a field of data (*byte*, *real*, *double*, or *integer*) and converts it to a *double* field. This may be useful for computing fields at greater data resolutions.

B.15 field to float cm

Transform any field to a field of single-precision floating point values.

Type filter

Inputs field field *any-dimension n-vector any-data any-coordinates*

Outputs field *same-dimension same-vector* float *same-coordinates*

Parameters	**Name**	**Type**	**Default**	**Choices**
	float normalize	toggle	off	on, off

The **field to float cm** module takes a field of data (*byte*, *short*, *real*, *double*, or *integer*) and converts it to a *float* field. It can be used in conjunction with modules that have a bias towards *float* fields.

B.16 field to int cm

Transform any field to an integer-valued field.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates*

Outputs field *same-dimension same-vector* integer *same-coordinates*

Parameters	**Name**	**Type**	**Default**	**Choices**
	int normalize	toggle	on	on, off

The **field to int cm** module takes a field of data (*byte*, *short*, *real*, *double*, or *int*) and converts it to an *int* field. This may be useful for performing integer math with greater precision than that offered by byte fields.

B.17 luminance cm

Compute the luminance of an image.

Type filter

Inputs field 2D uniform 4-vector byte (*image*)

Outputs field 2D uniform scalar byte

Parameters none

The **luminance cm** module computes the luminance (brightness) of an image, then outputs a 2-dimensional field of the same dimensions, but with a *scalar* byte value for each pixel in the original image instead of the full four-byte alpha, red, green, blue vector.

The luminance byte value can be used to produce a black and white version of the original image, or substituted back into the alpha byte of the original image to produce transparency effects.

B.18 orthogonal slicer cm

Slice through 3D or 2D field with plane perpendicular to coordinate axis.

Type mapper

Inputs field 3D or 2D *n-vector any-data any-coordinates*

Outputs field 2D or 1D *n-vector same-data same-coordinates*

Parameters

Name	Type	Default	Min	Max	Choices
slice plane	int	0	0	255	on, off
axis	choice	K			I,J,K

The **orthogonal slicer cm** module takes a 2D slice from a 3D array, or a 1D slice from a 2D array. It does so by holding the array index in one dimension constant, and letting the other index(es) vary.

B.19 threshold cm

Restrict values in data field.

Type filter

Inputs field *any-dimension n-vector any-data any-coordinates*

Outputs field of same type as input

Parameters

Name	Type	Default	Min	Max
thresh_min	float	0.0	none	none
thresh_max	float	255.0	none	none

The **threshold cm** module transforms the values of a field as follows:

- Any value less than the value of the **thresh_min** parameter is set to 0.

- Any value greater than the value of the **thresh_max** parameter is set to 0.
- Values within the **thresh_min**-to-**thresh_max** range are not changed.

After being **threshold**'ed, a data set's values are all either zero, or in this range:

thresh_min <= *value* <= **thresh_max**