NAME

specgram − create spectrograms from raw files or standard input

SYNOPSIS

specgram [**−aehlqvz**] [**−−print_input**] [**−−print_fft**] [**−−print_output**] [**−i, --input**=*RATE*] [**−r, --rate**=*RATE*] [**−d, --datatype**=*DATA_TYPE*] [**−p, --prescale**=*PRESCALE_FACTOR*] [**−b, --block_size**=*BLOCK_SIZE*] [**−f, --fft_width**=*FFT_WIDTH*] [**−g, --fft_stride**=*FFT_STRIDE*] [**−n, --window_function**=*WIN_FUNC*] [**−m, --alias**=*ALIAS*] [**−A, --average**=*AVG_COUNT*] [**−w, --width**=*WIDTH*] [**−x, --fmin**=*FMIN*] [**−y, --fmax**=*FMAX*] [**−s, --scale**=*SCALE*] [**−c, --colormap**=*COLORMAP*] [**--bg-color**=*BGCOLOR*] [**--fg-color**=*FGCOLOR*] [**−k, --count**=*COUNT*] [**−t, --title**=*TITLE*] *[outfile]*

DESCRIPTION

specgram generates nice looking spectrograms from raw data, based on the options provided in the command line.

The program has two output modes: file output when *outfile* is provided and live output when **−l, −−live** is provided. The two modes are not necessarily mutually exclusive, but behaviour may differ based on other options.

The program has two input modes: file input when the **−i, −−input** option is provided, or stdin input otherwise (default behaviour).

In file input mode, the file is read in a synchronous manner until EOF is reached, and the spectrogram is generated into *outfile*. Only file output is allowed in this mode, so *outfile* is mandatory and **−l, −−live** is disallowed.

In stdin input mode, data is read in an asynchronous manner and for an indefinite amount of time. The spectrogram is updated as new data arrives and output is buffered in memory.

In either input modes, when receiving SIGINT (i.e. by user pressing CTRL+C in the terminal), the program stops listening to data and exits gracefully, writing *outfile* if provided. This also happens in live output mode, when the live window is closed. If the program receives SIGINT again it will forcefully quit.

See **EXAMPLES** for common use cases.

OPTIONS

outfile Optional output image file. Check *SFML* documentation for supported file types, but PNG files are recommended.

If "**-**" is provided then the resulting image is written to stdout in PNG format.

Either *outfile* must be specified, **−l, −−live** must be set, or both.

−h, **−−help**
Display help message.

−v, **−−version**
Display program version.

INPUT OPTIONS

−i, **−−input**=*INFILE*
Input file name. If option is provided, *INFILE* is handled as a raw dump of values (i.e. input file format is not considered). The program will stop when EOF is encountered.

If option is not provided or "**-**" is provided, data will be read indefinitely from stdin.

−r, **−−rate**=*RATE*
Rate, in Hz, of the input data. Used for display purposes and computation of other parameters. Program will not perform rate limiting based on this parameter and will consume data as fast as it is available on stdin.

Default is 44100.

−d, **−−datatype**=*DATA_TYPE*
Data type of the input data. Is formed from an optional complex prefix (*c*), a type specifier (*u* for unsigned integer, *s* for signed integer, *f* for floating point) and a size suffix (in bits: 8, 16, 32, 64).

Valid values are: u8, u16, u32, u64, s8, s16, s32, s64, f32, f64, cu8, cu16, cu32, cu64, cs8, cs16, cs32, cs64, cf32, cf64.

Complex types are pairs of two values containing the real and imaginary part of the number, in this order. The size of the complex data type is twice that of the basic type. For example cf64 is 128-bit wide, corresponding to two 64-bit values.

Default is s16.

−p, **−−prescale**=*PRESCALE_FACTOR*
Input prescale factor.

The following normalizations are applied to input values, regardless if they are part of a complex number or not:
- unsigned values are normalized to [0.0 .. 1.0] based on the domain limits.
- signed values are normalized to [-1.0 .. 1.0] based on the domain limits.
- floating point values are left untouched, with the exception of NaN which is converted to a zero.

After this normalization, the new value is multiplied by *PRESCALE_FACTOR*. This is mostly useful for adjusting your inputs to the scale, and is usually needed for floating point inputs (see **−s, −−scale**).

Default is 1.0.

−b, **−−block_size**=*BLOCK_SIZE*
Block size, in data type sized values, that are to be read at a time from stdin. The larger this value, the larger the latency of the live spectrogram.

Default is 256.

FFT OPTIONS

−f, **−−fft_width**=*FFT_WIDTH*
FFT window width. Lower values provide worse frequency resolution but better temporal resolution. Higher values provide better frequency resolution but worse temporal resolution.

Default is 1024.

−g, **−−fft_stride**=*FFT_STRIDE*
Stride (distance) between two subsequent FFT windows in the input. Value can be less than *FFT_WIDTH* in which case there is overlap between windows, larger than *FFT_WIDTH* in which case information is lost, or equal to *FFT_WIDTH*.

Default is 1024.

−n, **−−window_function**=*WIN_FUNC*
Window function to be applied to the input window before FFT is computed. Because of the discrete nature of the FFT, a periodic assumption is made of the input window. In reality the input window is mostly never periodic, so window functions are used to taper off the ends of the window and avoid jumps between the beginning and end samples.

Valid values are: none, hann, hamming, blackman, nuttall.

Default is hann.

−m, **−−alias**=*ALIAS*
Specifies whether aliasing between negative and positive frequencies exists. If set to true (*1*), then the bins of corresponding negative and positive frequencies are summed on both sides.

Default is *0* (no) for complex data types and *1* (yes) otherwise.

−A, **−−average**=*AVG_COUNT*
Number of windows to average before the mean is displayed.

Use this for high sample rate signals, where either displaying many windows or computing too wide a window is not possible.

Default is 1.

DISPLAY OPTIONS

−q, **−−no_resampling**
Disables resampling of output FFT windows, generating clean and crisp output. This invalidates the use of **−w, −−width**, as the actual display width is computed from other parameters.

−w, **−−width**=*WIDTH*
Display width of spectrogram. Output FFT windows are resampled to this width, colorized and displayed. Cannot be used with**−q, −−no_r esampling**.

Default is 512.

−x, **−−fmin**=*FMIN*
Lower bound of the displayed frequency spectrum, in Hz.

Default is -*RATE*/2 for complex data types, 0 otherwise.

−y, **−−fmax**=*FMAX*
Upper bound of the displayed frequency spectrum, in Hz.

Default is *RATE*/2.

−s, **−−scale**=*SCALE*
Spectrogram scale, specified with the following format: *unit*[,*lower*[,*upper*]]

unit is an arbitrary string representing the unit of measurement (e.g. **V**). *lower* is an optional numeric value representing the lower bound of the scale. *upper* is an optional numeric v alue representing the upper bound of the scale.

Valid values for *SCALE* specify either just the unit, the unit and the lower bound, or all three values.

After normalization and prescaling (see **−p, −−prescale**), the following transformations are applied to the input:
- if*unit* starts with "dB", then a log arithmic decibel scale is assumed: Y=20*log10(X)
- the values are clamped between *lower* and *upper*: Y=clamp(X, *lower*, *upper*)

Default is dBFS,-120,0.

[dBFS] NOTE: The peak amplitude assumed for dBFS, after normalization and prescaling (see **−p, −−prescale**), is 1.0. Thus, the correct input domains are:
- [0 .. TYPE_MAX] for real unsigned integer values
- [-TYPE_MAX .. TYPE_MAX] for real signed integer values
- [-1.0 .. 1.0] for real floating point values
- { x | abs(x) <= TYPE_MAX } for complex signed and unsigned integer values
- { x | abs(x) <= 1.0 } for complex floating point values

Input values outside these domains may lead to positive dBFS values, which will be clamped to zero. Use prescaling (**−p, −−prescale**) to adjust your input to this domain. Integer inputs don't usually need prescaling, as they are normalized based on their domain's limits.

−c, **−−colormap**=*COLORMAP*
Color scheme. Valid values are: jet, gray, purple, blue, green, orange, red.

If *COLORMAP* is neither of these values, then it is interpreted either as a 6 character hex string (RGB color) or an 8 character hex string (RGBA color). In this case, a gradient between the background color and the color specified by the hex string will be used as a color map.

Default is jet.

−−bg-color=*BGCOLOR*
Background color. Either a 6 character hex string (RGB color) or an 8 character hex string (RGBA color).

Default is 000000 (black).

−−fg-color=*FGCOLOR*
Foreground color. Either a 6 character hex string (RGB color) or an 8 character hex string (RGBA color).

Default is ffffff (white).

−a, **−−axes**
Displays axes.

−e, **−−legend**
Displays legend. Entails **−a, −−axes**.

This is enabled in live view, but only for the live window (i.e. if both live view and file output are used, then file output will only display a legend if this flag is set by the user).

−z, **−−horizontal**
Rotates histogram 90 degrees counter clockwise, making it readable left to right.

−−print_input
Prints input windows to standard output, after normalization and prescaling (see **−p, −−prescale**).

−−print_fft
Prints FFT result to standard output, in FFTW order (i.e. freq[k] = *RATE**k/N).

−−print_output
Prints output, before colorization, to standard output. Values are in the domain [0.0 .. 1.0].

The length of the output may be different than the FFT result or the input, depending on specified frequency bounds (see **−x, −−fmin** and **−y, −−fmax**). Negative frequencies precede positive frequencies.

LIVE OPTIONS

−l, **−−live**
Displays a live rendering of the spectrogram being computed.

Either this flag must be set, *outfile* must be specified, or both.

−k, **−−count**=*COUNT*
Number of FFT windows displayed in live spectrogram.

Default is 512.

−t, **−−title**=*TITLE*
Title of live window.

Default is 'Spectrogram'.

EXAMPLE

One of the most obvious use cases is displaying a live spectrogram from the PC audio output (you can retrieve *yourdevice* using "**pactl list sources short**"):

parec --channels=1 --device="*yourdevice*.monitor" --raw | **specgram** -l

This will assume your device produces 16-bit signed output at 44.1kHz, which is usually the case.

If you want the same, but wider and with a crisp look:

parec --channels=1 --device="*yourdevice*.monitor" --raw | **specgram** -lq -f 2048

If you also want to render it to an output file:

parec --channels=1 --device="*yourdevice*.monitor" --raw | **specgram** -lq -f 2048 *outfile.png*

Keep in mind that when reading from stdin (like the above cases), the program expects SIGINT to stop generating FFT windows (e.g. by pressing CTRL+C in terminal). The file *outfile.png* will be generated after SIGINT is received.

Generating from a file to a file, with axes displayed and a crisp look:

specgram -aq -f 2048 -i *infile outfile.png*

Generating from a file to a file, with axes and legend displayed, but zooming in on the 2-4kHz band:

specgram -e -f 2048 -x 2000 -y 4000 -i *infile outfile.png*

Render a crisp output with a transparent background, so it can be embedded in a document:

specgram -qe --bg-color=00000000 -i *infile outfile.png*

Generating from a file to stdout and displaying the output with **imagemagick**:

specgram -i *infile* - | **display**

BUGS

Frequency bounds (**−x, −−fmin** and **−y, −−fmax**) may exceed FFT window frequency limits when resampling is enabled (i.e. default behaviour), but may not do so when resampling is disabled (**−q, −−no_resampling**). This inconsistency is known behaviour and, while not necessarily nice, does not impact usability in a meaningful manner. Ideally exceeding these limits should be allowed in both cases, and zero padding should be performed.

Moreover, when using the **−q, −−no_resampling** flag, the frequency limits are ±*RATE**(*FFT_WIDTH*-1)/(2**FFT_WIDTH*) when *FFT_WIDTH* is odd and -*RATE**(*FFT_WIDTH*-2)/(2**FFT_WIDTH*) to *RATE*/2 when *FFT_WIDTH* is even. This is a bit different from the behaviour of NumPy's implementation of fftfreq and aims to make it easier to display the Nyquist frequency component for non-complex inputs.

The above upper limits are enforced silently in the default values of **−x, −−fmin** and **−y, −−fmax**, but for brevity are not mentioned in this manpage's **OPTIONS** section or in the program help screen.

AUTHORS

ACKNOWLEDGEMENTS

Taywee/args library by Taylor C. Richberger and Pavel Belikov, released under the MIT license.

Program icon by Flavia Fabian, released under the CC-BY-SA 4.0 license.

Share Tech Mono font by Carrois Type Design, released under Open Font License.

Special thanks to Eugen Stoianovici for code review and various fixes.