Mali Texture Compression Tool v4.3.0

User Guide

ARM® MALI™

Visual Computing

malideveloper.arm.com

Contents

1 Introduction

The Mali Texture Compression Tool allows you to convert texture image files from their native format into Ericsson Texture Compression Version 1, Ericsson Texture Compression Version 2, and Adaptable Scalable Texture Compression format. Compressing your texture data allows your application to save texture bandwidth and hence increase performance and lower power consumption.

The Mali Texture Compression Tool can read and convert many common image formats including `bmp`, `jpg`, `png`, `tga`, `gif`, `tif` and `psd`. It can also uncompress `ktx` and `pkm` files to a variety of common formats, including `ppm`, `pgm`, `png`, `gif`, `bmp`, `tif` `tiff`, `psd`, `tga`, `raw`, `pct`, `sgi` and `xpm`. The Mali Texture Compression Tool supports high dynamic range textures in `dds`, `hdr` and `pic` formats.

Support for compressing Adaptable Scalable Texture Compression encoded 3D textures is available for uncompressed `ktx` and `dds` files. The Mali Texture Compression Tool does not support Ericsson Texture Compression format compression of 3D textures, nor does it support reading cube-map, nor 2D-array images.

2 Minimum Requirements

The Mali Texture Compression Tool ships with the necessary dependencies to run, however if you experience problems with compression, particularly on Linux systems, you may be missing a library dependency for the compression / image conversion tools used by the UI. See Known Issues for more information.

Using compressed textures in your application requires appropriate support from your graphics drivers.

3 Installation and Configuration

The Mali Texture Compression Tool consists of two components: a GUI application and a set of command-line binaries. The command-line binaries are used to perform the compression and decompression of texture data. The GUI is a wrapper around these binaries that allow you to view the results of compression, including the ability to view a visual diff of the source and compressed image.

The installer packages for each platform contain both the GUI and the command-line binaries.

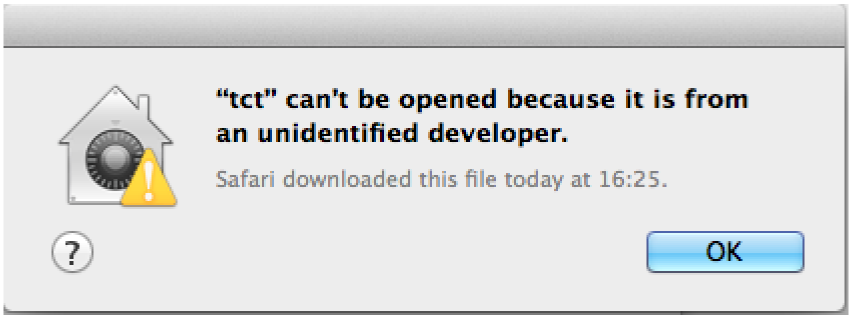
3.1 Linux and Mac OS X

To install on Linux and Mac OS X, simply extract the downloaded tarball to a location on disk. The installation package for Linux and Mac OS X is self-contained and no further installation is necessary.

The command-line binaries can be found in the `bin` subdirectory, and the GUI can be launched by running the `tct` binary in the top-level of the installation.

3.1.1 Cannot open on OS X Mavericks

Due to changes in OS X Mavericks you may find you are not able to launch the Mali Texture Compression Tool due to the following warning:



"tct" can't be opened because it is from an unidentified developer.

Safari downloaded this file today at 16:25.

If you come across this issue, you can work around it by holding down the `ctrl` key and clicking on `tct` then selecting `Open` from the popup menu. You will then be prompted whether or not you want to continue to open the Mali Texture Compression Tool. You should click `Ok` to start the tool.

3.2 Windows

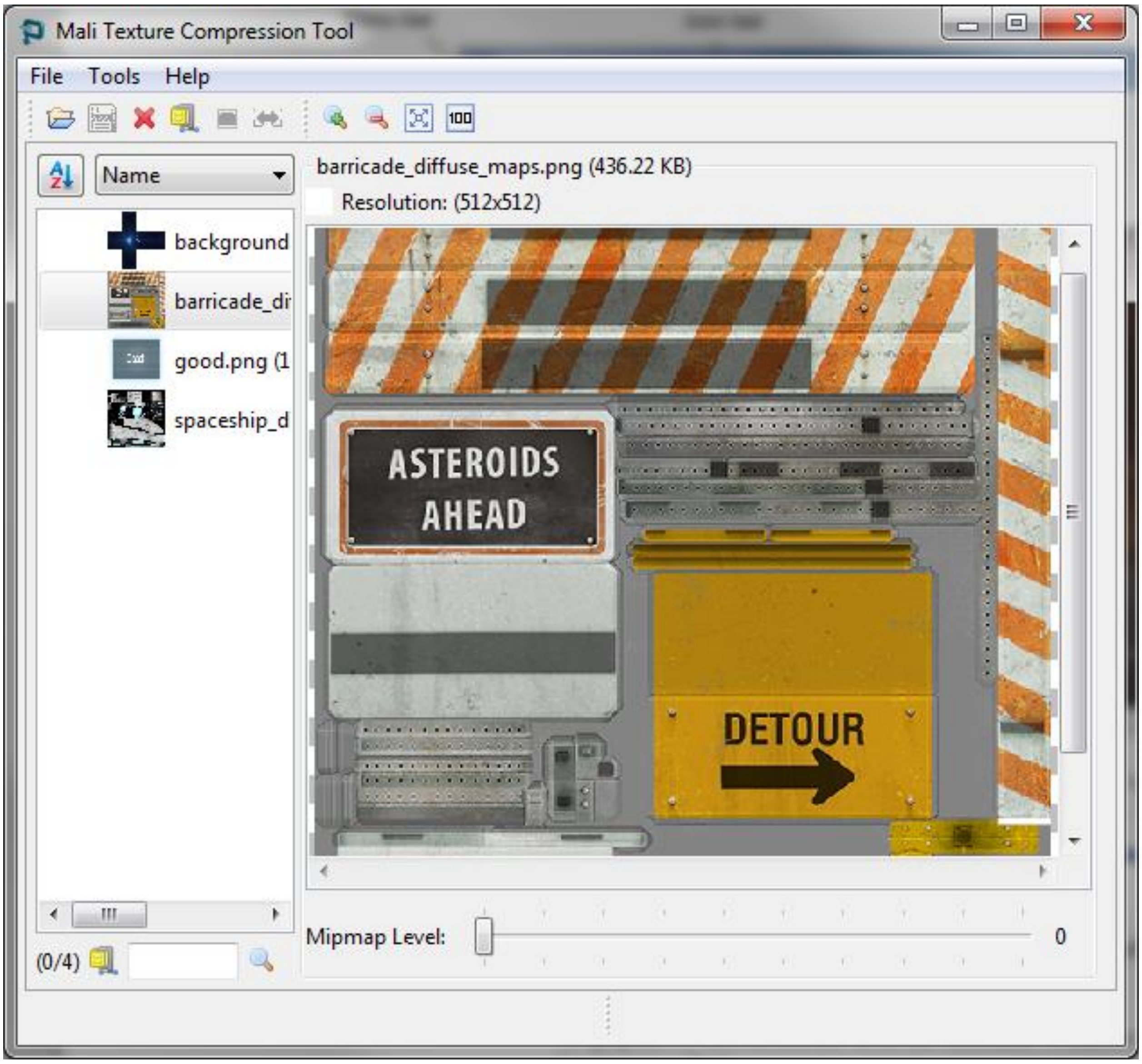
To install on Windows, run the downloaded installer package and follow the instructions on screen. By default, the installer will place the Mali Texture Compression Tool in the `Program Files\ARM\Mali Developer Tools\Mali Texture Compression Tool v4.3.0` directory.

The command-line binaries can be found in the `bin` subdirectory of this installation, and the GUI can be launched from the Windows Start Menu.

4 Getting Started with the Mali Texture Compression Tool

4.1 The Main Window

In use, the Mali Texture Compression Tool looks like the following screenshot (which is taken on Windows. There are minor differences in appearances on other operating systems):



Mali Texture Compression Tool Main Window

This window shows the Texture List on the left-hand side, and a texture preview in the main area. You can control the Mali Texture Compression Tool via either the menu items or the toolbar icons. Many functions are also available by right-clicking on different areas in the application.

4.1.1 The Texture List

The texture list shows the images you have loaded into the application. These images can be in many different formats, and may be compressed or uncompressed images. You can use the drop-down list box to control the sort order of the images in the list. To reverse the sort order, you can click the *Sort order* button next to the list.

Should you have a lot of texture images in the list, the text box at the bottom of the texture list can be used to filter the list by matching any entered text.

4.1.2 3D Texture Support

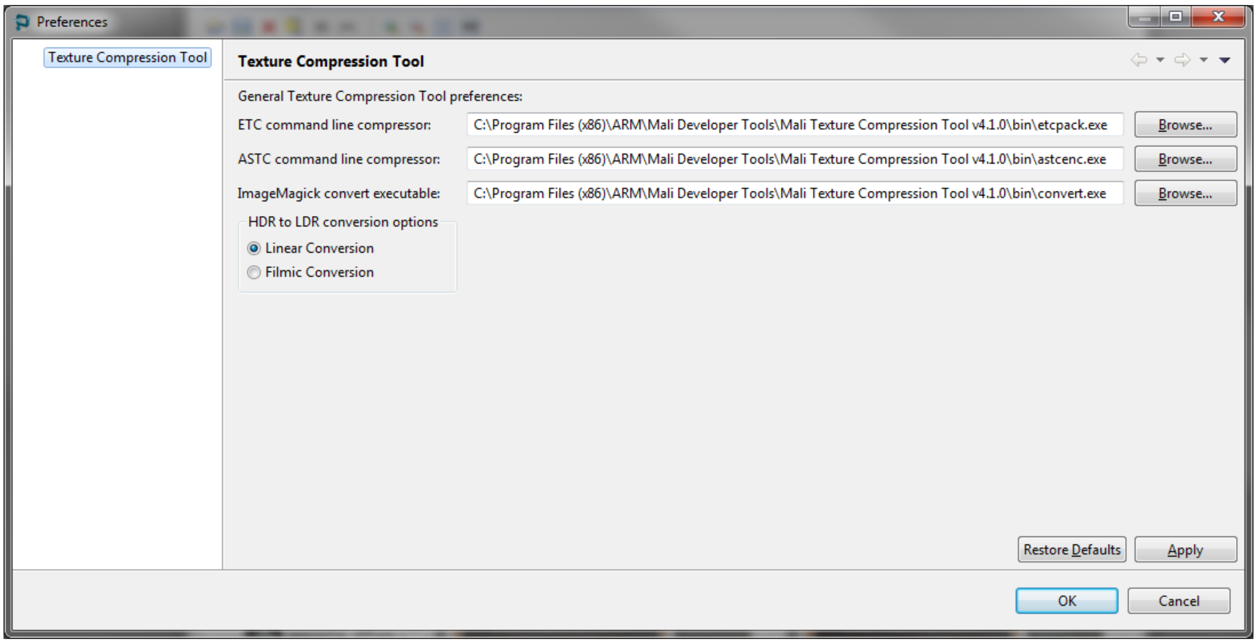
For display purposes, 3D textures are presented as an expandable list of 2D images, with one image representing each depth layer of the 3D image. Each depth image may have its own set of mipmaps listed beneath it. Users are able to compare the compressed output for each depth image, in much the same way they would compare a compressed 2D image using the Image Difference Viewer (see 4.5.1)

4.1.3 The Image Preview

The image preview pane in the main screen displays the image currently selected in the texture list. If the image has been compressed, both the original and compressed images are displayed. You can use the *Mipmap Level* slider at the bottom of the image preview to change the mip level currently displayed.

4.2 Setting the Location to the Command Line Binaries

The preferences window, accessed via *Tools > Preferences*, allows you to set the locations of the command-line binaries. It also allows you to set the method by which high dynamic range images will be converted to low dynamic range images.



General Texture Compression Tool preferences:

ETC command line compressor: C:\Program Files (x86)\ARM\Mali Developer Tools\Mali Texture Compression Tool v4.1.0\bin\etcpack.exe

ASTC command line compressor: C:\Program Files (x86)\ARM\Mali Developer Tools\Mali Texture Compression Tool v4.1.0\bin\astcenc.exe

ImageMagick convert executable: C:\Program Files (x86)\ARM\Mali Developer Tools\Mali Texture Compression Tool v4.1.0\bin\convert.exe

HDR to LDR conversion options

[✓ Linear Conversion]

Filmic Conversion

Mali Texture Compression Tool

Before you can compress textures using the GUI, you must configure the locations of `etcpack`, `convert` and `astcenc`. By default, these should be configured to the correct locations, but if not then you can configure the locations in the Preferences pane of the GUI.

4.3 Loading an Image into the GUI

To load an image into the GUI, use *File > Open images...*, or click on the *Open images...* button on the toolbar. You can select as many files in as many different formats as you like, provided those formats are supported by the GUI. For a complete list of supported formats, see the *ImageMagick supported formats page*.

4.4 Compressing an Image

To compress an image, select it in the image list on the left and select *File > Compress selected images...*, or click on the *Compress selected images...* button on the toolbar. You will see a dialog where you can select the compression options you would like to use.

4.4.1 Selecting a Compression Format

The Mali Texture Compression Tool can compress textures in ETC1, ETC2 and ASTC format. The format you use and the specific compression options you select will depend largely on your requirements. In general:

- ASTC is the highest-quality compression but less widely-supported,
- ETC1 is the lowest-quality compression but most widely supported,
- ETC2 is a good tradeoff between compression quality and breadth of support.

Note also that ETC1 is a standard format for OpenGL ES 2.0 implementations and ETC2 is a standard format for OpenGL ES 3.0 implementations.

4.4.2 Compression Profiles

The Mali Texture Compression Tool allows you to save and recall different compression options in profiles that can be applied to subsequent compressions. To store the currently selected settings, click the *Save profile* button next to the drop-down list and enter a name for the profile. To recall a profile, select it from the drop-down list.

Note: Profiles for ASTC and ETC1/ETC2 are stored separately and are not transferable.

4.4.3 Setting an Output Directory

The output directory is used to store the results of compressing texture images. To set the location for output, click the *Select output directory* button and choose a location. Images will be saved in that location.

4.4.4 Compressing to ETC1 or ETC2 Format

To compress to ETC1 or ETC2 format, select the *ETC1/ETC2* tab in the compression options dialog.

Output File Format

The output file format determines the format of the output file that will be generated. The choices are `PKM` or `KTX` format.

- `PKM` is a simple file format for single compressed images. If you generate Mipmaps, multiple `PKM` files are created.
- `KTX` is a format that provides a container for images. If you generate Mipmaps, a single `KTX` file is generated.

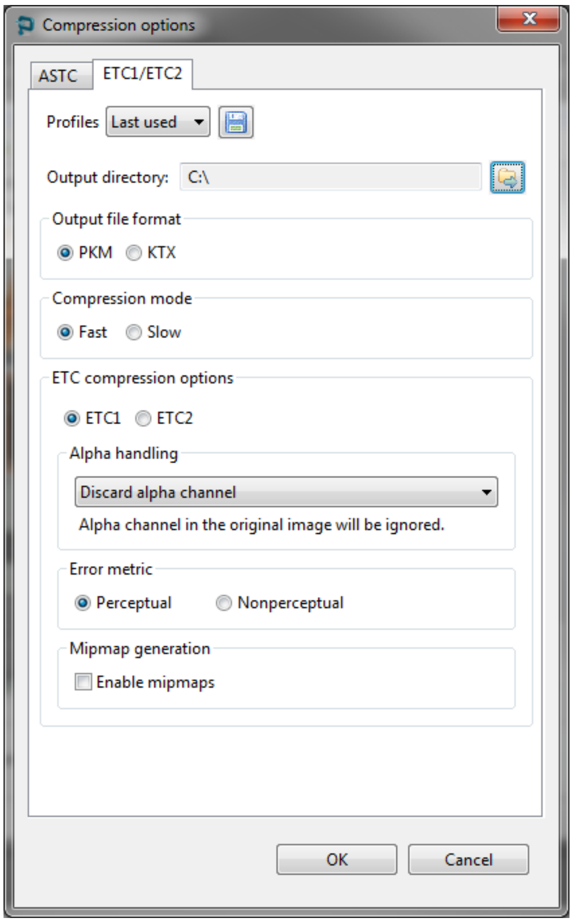
Compression Mode

The compression mode determines how thoroughly the ETC algorithm will process the image data.

- Fast mode gives fast compression times, but at the expense of image quality.
- Slow mode gives higher image quality, but takes more processing time.

ETC1 Compression

ETC1 is compatible with OpenGL ES 1.1 and higher. If ETC1 output is selected, the following options are available:



Profiles: Last used

Output directory: C:\

Output file format: [✓ PKM] KTX

Compression mode: [✓ Fast] Slow

ETC compression options: [✓ ETC1] ETC2

Alpha handling: Discard alpha channel

Alpha channel in the original image will be ignored.

Error metric: [✓ Perceptual] Nonperceptual

Mipmap generation: ☐ Enable mipmaps

ETC1/ETC2 Compression Options

Alpha Handling

This setting determines how the alpha channel is handled. ETC1 does not support images with alpha channels, so you are able to select one of four options:

- *Discard Alpha Channel* simply discards the original image's alpha channel.
- *Create Atlas* creates a combined image containing an ETC1-compressed alpha channel image followed by an ETC1-compressed colour image.
- *Create separate compressed image* creates a separate ETC1-compressed image containing the alpha channel.
- *Create separate uncompressed image* copies the alpha channel to a separate file without compression.

Error Metric

The error metric determines how the ETC algorithm handles the Peak Signal-to-Noise Ratio (PSNR) in the output image.

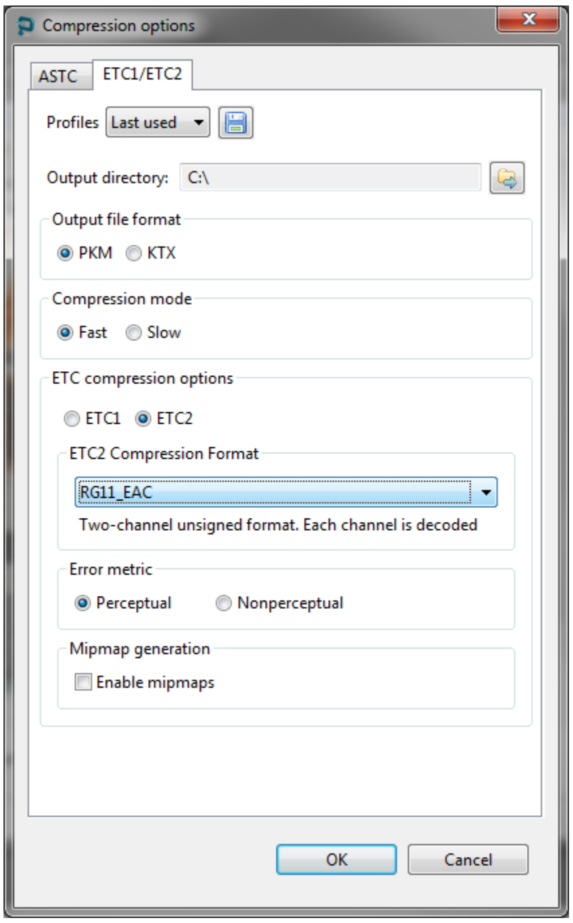
- *Perceptual* mode provides the best visual results. Using Perceptual, the compression algorithm sets green closer to its required value, at the expense of an inferior representation of red and blue. This decreases the Peak Signal-to-Noise Ratio (PSNR), but provides a superior visual result because the eye is more sensitive to green than to blue and red.
- *Non-perceptual* mode is optimised to provide the highest PSNR. This is the default. Although technically superior, this setting does not account for the fact that the eye is more sensitive to green than to blue and red, and images might appear to be visually inferior.

Enable Mipmaps

If this checkbox is selected, mip levels down to 1x1 will be created.

ETC2 Compression

ETC2 is compatible with OpenGL ES 3.0. If ETC2 output is selected, the following options are available:



Compression options

ASTC | ETC1/ETC2

Profiles: Last used

Output directory: C:\

Output file format: [✓ PKM] KTX

Compression mode: [✓ Fast] Slow

ETC compression options: ETC1 [✓ ETC2]

ETC2 Compression Format: RG11_EAC

Two-channel unsigned format. Each channel is decoded

Error metric: [✓ Perceptual] Nonperceptual

Mipmap generation: ☐ Enable mipmaps

OK | Cancel

ETC1/ETC2 Compression Options

ETC2 Compression Format

The ETC2 compression format drop-down list determines the exact ETC2 format to use for compression. The options are:

- `RGB8_ETC2` Compress RBG8 data with no alpha channel.
- `RGBA8_ETC2_EAC` Encodes RGBA8 data. The RGB part is encoded the same as `RGB_ETC2`, but the alpha part is encoded separately.
- `R11_EAC` One-channel (red) unsigned format. It is similar to the alpha part of `RGB8_ETC2_EAC` but delivers higher precision.
- `RG11_EAC` Two-channel (red and green) unsigned format. Each channel is decoded exactly as with `R11_EAC` encoding. * `SIGNED_R11_EAC` One-channel signed format. It allows preserving zero exactly, while still using both positive and negative values.
- `SIGNEDRG11_EAC` Two-channel (red and green) signed format. Each channel is decoded exactly as for `SIGNED_R11_EAC`.
- `RGB8_PUNCHTHROUGH_ALPHA1_ETC2` This format is very similar to `RGB8_ETC2`, but has the ability to represent punchthrough alpha, which is the ability to make it completely opaque or transparent.

Error Metric

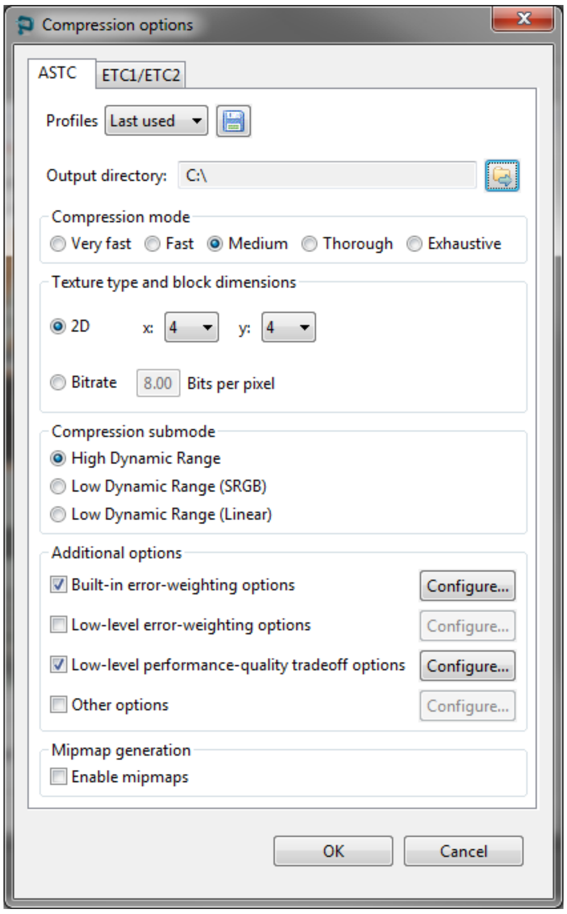
The error metric for ETC2 is handled in exactly the same way as the error metric for ETC1.

Enable Mipmaps

If this checkbox is selected, mip levels down to 1x1 will be created.

4.4.5 Compressing to ASTC Format

To compress to ASTC format, click the *ASTC* tab in the compression options dialog. The following options are available:



Compression options

ASTC | ETC1/ETC2

Profiles: Last used

Output directory: C:\

Compression mode: Very fast / Fast / [✓ Medium] / Thorough / Exhaustive

Texture type and block dimensions:
[✓ 2D] x: 4 y: 4
Bitrate 8.00 Bits per pixel

Compression submode:
[✓ High Dynamic Range]
Low Dynamic Range (SRGB)
Low Dynamic Range (Linear)

Additional options:
[✓ Built-in error-weighting options] Configure...
Low-level error-weighting options Configure...
[✓ Low-level performance-quality tradeoff options] Configure...
Other options Configure...

Mipmap generation:
Enable mipmaps

OK Cancel

ASTC Compression Options

Compression Mode

The compression mode determines how thoroughly the ASTC codec will process the input image. Each compression mode represents a tradeoff of compression speed and final image quality.

Texture Type and Block Dimensions

The texture type and block dimensions settings control the way that the ASTC algorithm will select what block size to use when compressing. It's possible to set either an explicit block size, or to set a desired target bitrate and have the algorithm choose an appropriate block size.

- When setting an explicit block size, the valid dimensions are `4`, `5`, `6`, `8`, `10` or `12` for 2D images, and `3`, `4`, `5`, or `6` for 3D images.
- When setting a target bitrate, block dimensions are chosen to most closely match that bitrate. The bitrate should be a decimal value. For example, to specify a bitrate of 2 bits per texel, enter `2.0`. When specifying a bitrate that could potentially map to many block dimensions, the algorithm will pick the block dimensions that produce the most square

aspect ratio (for example, a bitrate of `2.67` will cause the algorithm to pick a block size of `8x6` rather than `12x4`).

Compression Submode

This option controls how the input image will be treated by the ASTC algorithm.

- *High Dynamic Range* specifies that the input image should be treated as a high dynamic range image.
- *Low Dynamic Range (SRGB)* specifies that the input image should be treated as a low dynamic range image in a sRGB colour space.
- *Low Dynamic Range (Linear)* specifies that the input image should be treated as a low dynamic range image in a linear colour space.

Additional Options

This section controls low-level ASTC options. To use them, enable the check box and click *Configure...* to control specific options.

Built-In Error-Weighting Options

These options provide extra combinations of settings that configure the ASTC algorithm to handle a variety of common texture types.

- *Apply PSNR Optimization*: For encoding, assume that the input texture is a normal map with the X and Y components of the actual normals in the red and green colour channels. The algorithm will then move the 2nd component to alpha, and apply an error-weighting function based on angular error. It is possible to use this preset with texture decoding as well, in which case it will expand the normal map from 2 to 3 components after the actual decoding.
- *Apply perceptual optimization*: Similar to *Apply PSNR Optimization*, except that the algorithm will try to optimise the normal map for best possible perceptual results instead of just maximising angular PSNR.
- *Apply mask optimization*: Assume that the input texture is a texture that contains unrelated content in its various colour channels, and where it is undesirable for errors in one channel to affect the other channels.
- *Apply alpha blending optmization*: Assume that the input texture is an RGBA texture where the alpha component is used to represent opacity.
- *Apply HDR optimization*: Assume that the input texture is an HDR texture. If an alpha channel is present, it is treated as an LDR channel (e.g. opacity). Optimise for 4th-root error for the colour and linear error for the alpha.
- *Apply HDR with alpha channel optimization*: Assume that the input texture is an HDR texture, and optimise for 4th-root error. If an alpha channel is present, it is assumed to be HDR and optimised for 4th-root error as well.
- *Apply HDR for logarithmic error optimization*: Assume that the input texture is an HDR texture, and optimise for logarithmic error. This should give better results than *Apply HDR optimization* on metrics like "logRMSE" and "mPSNR", but the subjective quality (in particular block artifacts) is generally significantly worse.
- *Apply HDR with alpha channel for logarithmic error optimization*: As above, with alpha channel support.

Low-Level Error-Weighting Options

These options provide low-level control of the error weighting of the ASTC algorithm.

- *Per-texel relative error weighting for the RGB color channels*: Compute per-texel relative error weighting for the RGB colour channels as follows:

```
weight = 1 / (<base weight> + <avg scale>) * average ^ 2 + <stdev scale>
* stddev ^ 2
```

where `average` and `stdev` are computed as the average-value and the standard deviation across the neighbourhood of each texel; the *Radius* controls how wide this neighbourhood should be.

- *Per-texel relative error weighting for the RGBA color channels*: As above, with alpha channel support.
- *Scale per-texel weights by alpha*: For textures with alpha, scale the per-texel weights by the alpha value. The value chosen for scaling of a particular texel is taken as an average across the neighbourhood of the texel, where *Radius* controls the area of the sampling. A *Radius* of `0` causes the texel's own alpha value to be used, with no contribution from neighbouring texels.
- *Relative weight to each colour channel*: Assign a relative weighting to each colour channel. If this option is combined with any other options, those options are used to calculate a weighting, then that weighting is multiplied by this value.
- *Scale error weighting to match angular error*: Assume that the red and alpha channels (after any swizzling) represent the X and Y components for a normal map, and scale the error weighting so as to match the angular error as closely as possible. The reconstruction function for the Z component is assumed to be `Z = sqrt(1 - X ^ 2 - X ^ 2)`.
- *Increase error weight for edges and corners*: Increase the error weight for texels at the compression-block edges and corners. The value specifies how much the weights are to be modified, with `0` giving no modification. Higher values should reduce block-artifacts, at the cost of worsening other artifacts.

Low-Level Performance-Quality Tradeoff Options

These options provide low-level control of the performance-quality tradeoffs provided by the ASTC algorithm.

- *Partitioning limit*: Test only a given number of differing partitions. Higher numbers give better quality at the expense of a longer decoding time, however large values tend to give diminishing returns.
- *PSNR limit*: Stop compression work on a block as soon as the PSNR block exceeds the specified dB limit. Note that the algorithm is not guaranteed to reach the PSNR number for a given block. This option is ineffective for HDR textures.
- *Partitioning error limit*: Stop compression work when the error from encoding with 2 partitions exceeds that of encoding with 1 partition by the given amount. The algorithm will not apply this factor to normal map inputs.
- *Correlation coefficients limit*: For each block, the algorithm will compute the correlation coefficients between any two colour components; if no pair of colours have a correlation coefficient below the cutoff specified, the algorithm will not try dual-weight-planes. If the input is a normal map, the algorithm will use a value of `0.99`.
- *Block modes cutoff limit*: Cutoff on the set of block modes to use; the cutoff is a percentile of the block modes that are most commonly used. This option takes a value from 0 to 100, where 0 offers the highest speed and lowest quality, and 100 offers the highest quality and lowest speed. This option is ineffective for HDR textures.
- *Refinement iterations limit*: Maximum number of refinement iterations to apply to colours and weights. Minimum value is 1; larger values give slight quality increase at expense of mild performance loss.

Other Options

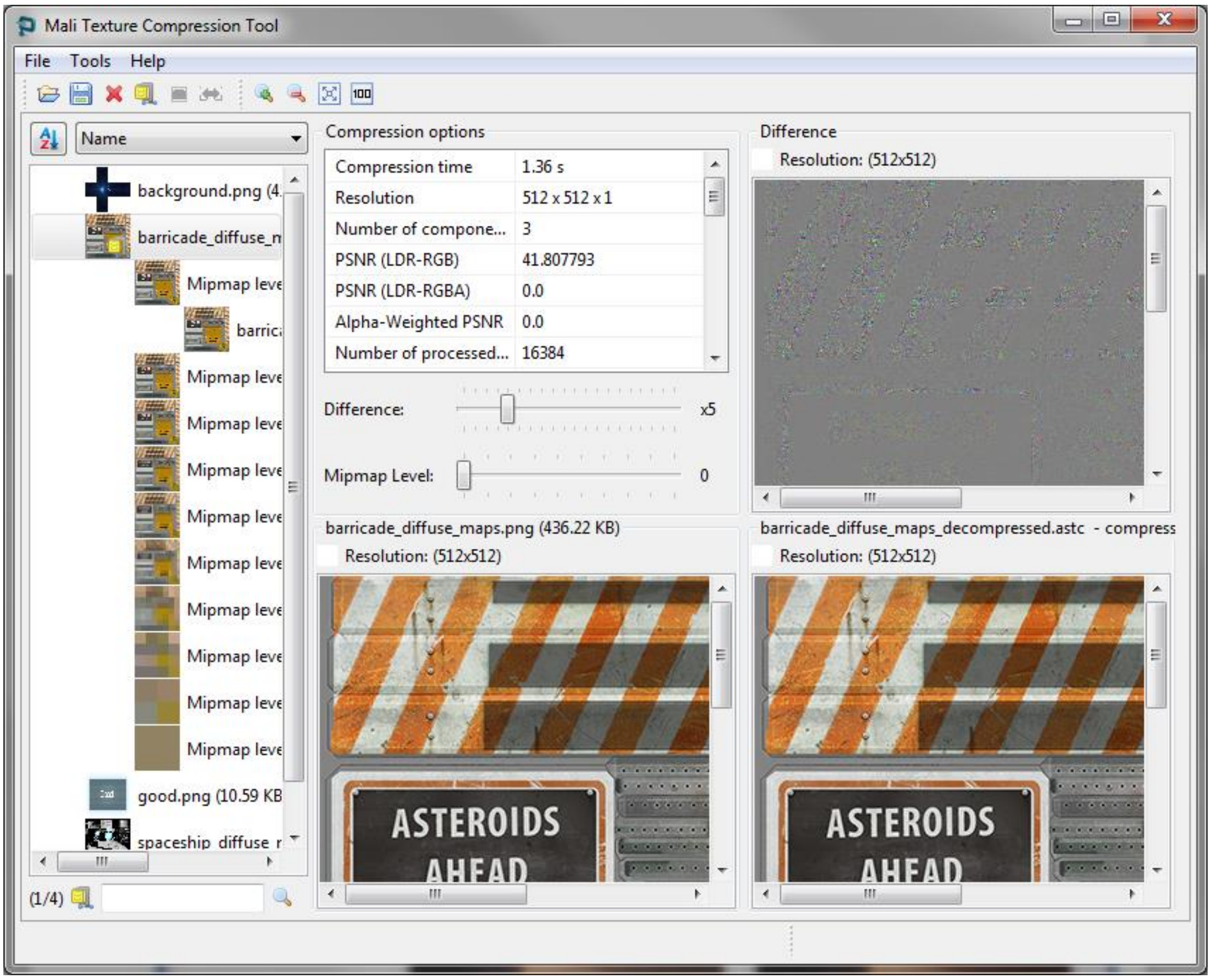
These options provide additional settings to the ASTC algorithm.

- *Force the use of HDR endpoint modes*: Force HDR output. If enabled, the radio buttons control how to handle the alpha channel.

- *Swizzle color components*: Swizzle the colour components either before or after encoding the image.
- *Signed-linear colour conversion*: Convert input image from sRGB to linear-RGB before encode; convert output image from linear-RGB to sRGB after decode. For encode, the transform is applied after swizzle; for decode, the transform is applied before swizzle.
- *Multithreading*: Force the number of threads to use for encoding. If this option is not enabled, the algorithm will spawn as many threads as there are CPU cores.
- *mPSNR error metric*: Sets low and high f-stop values to use for the mPSNR error metric. The default low value is `-10`, and the default high value is `10`. The mPSNR error metric applies only to HDR textures.

4.5 Working with Compressed Output

After compression, the main window changes to display four panes of information:



Compression options

Compression time	1.36 s
Resolution	512 x 512 x 1
Number of compone...	3
PSNR (LDR-RGB)	41.807793
PSNR (LDR-RGBA)	0.0
Alpha-Weighted PSNR	0.0
Number of processed...	16384

Difference: x5

Mipmap Level: 0

The main window after compression

If the *Enable mipmaps* option was specified, the mipmap levels are displayed in the texture list. In the case of 3D textures, each depth image will have its own set of mipmap images listed. Clicking on them will show the levels in the image preview pane. Selecting the main image, or a 3D depth image will show the image difference viewer.

4.5.1 The Image Difference Viewer

The image difference viewer shows the original image, the compressed output, a visual diff of those images and a table of information about the compressed image. By hovering the mouse over different pixels of the images, you can see the pixel values at that location.

To change the mip level currently displayed in the image, use the *Mipmap level* slider. The differences between the source and compressed images can also be amplified by using the *Difference* slider. This will make the differences between the two images easier to see.

5 Using the Mali Texture Compression Tool Command Line Binaries

Usage of the command-line binaries is considered an advanced topic and not described in this guide. For those wishing to use the binaries directly, consult the `--help` output of the executables.

6 Known Issues and Limitations

6.1 No Multithreading Support

The Texture Compression Tool does not support compressing multiple images concurrently. When multiple images are selected to be compressed, each will be queued and compressed sequentially.

The ASTC codec supports compressing a single image using multiple threads, so it is possible to gain some speed up during compression by selecting this option. You can enable threaded compression by selecting the "Other Options" button under "Additional Options" in the ASTC compression settings window, then selecting "Apply" next to the multithreading option.

6.2 Metric and Speed Options are not Available in all Compression Formats

The "metric" options (perceptual and non-perceptual) and "speed" options (fast and exhaustive) are not available for all types of compression formats in the current version. The table below shows how these options are supported in each compression format.

Codec	Format	Perceptual	Non-perceptual	Fast	Exhaustive
ETC1	`RGB`	Yes	Yes	Yes	Yes
ETC2	`R`	No	No	No	No
ETC2	`R_signed`	No	No	No	No
ETC2	`RG`	No	No	No	No
ETC2	`RG_signed`	No	No	No	No
ETC2	`RGB`	Yes	Yes	Yes	Yes
ETC2	`RGBA`	Only for RGB channels	Only for RGB channels	Yes	Yes
ETC2	`RGBA1`	No	Yes	Yes	No

6.3 Alpha Channel PSNR is Always Zero when Compressing with no Mipmaps

When compressing without mipmap generation, the alpha channel's PSNR value is always zero.

6.4 `.hdr` Images Cannot be Compressed when "Enable Mipmaps" and ASTC Compression Options are Selected

When attempting to compress `.hdr` images with the ASTC algorithm, mipmaps will not be generated and an error is not displayed.

6.5 Some `.tga` Imags Fail to Compress

An ETC compression on some `.tga` images will fail with the following error: "Color resolution must be 255". You may be able to compress the image by converting it to a different format and back again with an image editor.

6.6 Missing library dependencies on some Linux distributions

There are a number of shared libraries that are required for the image conversion tool that is used during image compression. If any of the libraries cannot be found, the Texture Compression Tool will be unable to complete compression jobs.

The compression tool will warn you on launch if it is unable to find the shared library in the system library folder(s).

6.6.1 Missing libtiff.so.4 on Ubuntu versions later than 12.04

The version of libtiff shipped with Ubuntu 14.04 and later is libtiff.so.5. There is no libtiff.so.4 providing package for these versions of Ubuntu. Instead you must use the package from Ubuntu 12.04 available from http://packages.ubuntu.com/precise-updates/libtiff4

7 Changes from Previous Versions

7.1 Changes in Version 4.3.0

- Bundle JRE with application. Java dependency updated to Java 8.
- Updated Eclipse dependency to Luna SR2.
- Fixed issue where unable to compress certain HDR files. [TCTOOL-834]
- Fixed issue where toolbar layout changes arbitrarily [TCTOOL-832]
- Fixed issue where Stop Compression button is sometimes unresponsive. [TCTOOL-831]
- Fixed issue where compressing to ASTC with swizzle parameters set fails to compress. [TCTOOL-823]
- Added known issue Missing library dependencies on some Linux distributions [TCTOOL-820]
- Fixed missing resolution value in Compression Options table. [TCTOOL-814]
- Fixed issue where the tool failed to compress image files with invalid non-ascii or punctuation characters in the name. [TCTOOL-806]
- Fixed issue loading certain KTX format files. [TCTOOL-794]
- Added known issue Some .tga Images Fail to Compress [TCTOOL-786]
- Drag & Drop works for folders now [TCTOOL-769]

7.2 Changes in Version 4.2.0

- Support for compressing uncompressed 3D textures.
- Fixed issue where certain KTX files are incorrectly read.
- Fixed issue where etcpack cannot be run concurrently.
- Fixed issue with 1/2/4-bit palette PNG images.

7.3 Changes in Version 4.1.0

- Support for Adaptable Scalable Texture Compression format.
- Support for high dynamic range textures.
- Support for compression profiles. Compression profiles allow you to save particular combinations of settings to be re-used in future compressions.

7.3.1 Changes in Version 4.0.1

- Fixed an issue where it was not possible to use the slow compression method.
- More accurate handling of 16-bit images for R and RG compression options
- The dependency on ImageMagick has been removed on Mac OS X.

7.3.2 Changes in Version 4.0.0

- Support for ETC2 compression formats:
  - `R (GL_EAC_R11_UNSIGNED_OES)`
  - `R_signed (GL_EAC_R11_SIGNED_OES)`
  - `RG (GL_EACX2_RG11_UNSIGNED_OES)`
  - `RG_signed (GL_EACX2_RG11_SIGNED_OES)`
  - `RGB (GL_ETC2_RGB8_OES)`
  - `RGBA and RGBA4 (GL_ETC2_RGBA8_OES)`
  - `RGB1 (GL_ETC2_PUNCHTHROUGHA_RGBA8_OES)`

7.3.3 Changes in Version 3.0.0

- Significant performance increases.
- Support for KTX output formats.
- Eclipse-based GUI.
- Support for a wider range of input formats.
- Support for image alpha channels.

- Support for opening previously compressed images.

8 Support

For support on this product, please visit the ARM Mali Graphics Community.

8.1 Continued Support

It should be noted that continuing support of the product will only be provided by ARM if such support is covered by a current contract with the recipient.

9 Legal

9.1 Proprietary Notice

9.2 Analytics Information

The following information is tracked by the Texture Compression Tool and sent to ARM for analytics purposes.

When Tracked	Name	Summary
Per compressed image	ASTC/astc_compression_bitrate	Tracks the bitrate associated with the ASTC compression job
Per compressed image	ASTC/astc_compression_speed	Tracks the speed category of the compression job (as defined in the ASTC compression settings)
Per compressed image	ASTC/astc_compression_submode	Tracks the ASTC compression submode
Per compressed image	ASTC/astc_compression_type	Tracks the bitrate associated with the ASTC compression job
Per compressed image	compression/compression_file_type	Tracks the file type of the image to be compressed
Per compressed image	compression/compression_outcome	Tracks whether the outcome succeeded or failed
Per compressed image	compression/image_compression_error	Tracks an error thrown during compression
Per compressed image	compression/mipmaps_generated	Tracks whether mipmaps have been enabled
Per compressed image	ETC/etc_compression_alpha	Tracks the alpha handling mode associated with an ETC1 job
Per compressed image	ETC/etc_compression_codec	Tracks whether the user has used ETC version 1 or 2
Per compressed image	ETC/etc_compression_error	Tracks the ETC compression error metric
Per compressed image	ETC/etc_compression_format	Tracks the ETC2 compression format
Per compressed image	ETC/etc_compression_out	Tracks the output format of an ETC compression job
Per compressed image	ETC/etc_compression_speed	Tracks ETC job speed setting

Per product version	product/build	Tracks the current product build ID
Per product version	product/version	Tracks the current product version
Per session	action/image_compression_jobs	Tracks the number of times a user has compressed an image
Per session	action/image_opened	Tracks the number of times the user opens a file
Per session	action/viewer_sort	Tracks the number of times the user changes the sort criteria for the texture list
Per session	action/viewer_sort_inverse	Tracks the number of times the user inverts the sort for the texture list
Per session	memory/amount_available	Tracks the amount of memory available to Java when the program is first run
Per session	product/user_preference	Tracks which user preferences are enabled and disabled
Per session	system_property/os_arch	Tracks the operating system bitness
Per session	system_property/os_distribution_description	(Linux only) Tracks the Linux distribution version descriptive name
Per session	system_property/os_distribution_name	(Linux only) Tracks the Linux distribution name
Per session	system_property/os_distribution_version	(Linux only) Tracks the Linux distribution version number
Per session	system_property/os_name	Tracks the operating system name
Per session	system_property/os_version	Tracks the operating system version
Per session	system_property/user_country	Tracks the users locale country
Per session	system_property/user_language	Tracks the users locale language

To opt out of analytics tracking please choose the option in the Preferences window.