Matlas_tools Toolbox: A mex Gateway for Utilizing lastools Read and Write Functions in MATLAB

Authors

Paula Litkey has written most of the code of *matlas_tools* and documentation for using the toolbox in Linux.
Eetu Puttonen has contributed to the *matlas_tools* coding and testing. He has written the documentation for compiling and using the toolbox in Windows.

The authors welcome all comments for improving the toolbox and its documentation.

Acknowledgements

Martin Isenburg, for creating, developing, and maintaining the LAStools and LASlib, http://lastools.org

Joaquim Luis, for providing a test mex code, header files for .dll compilation, and MATLAB compatible dynamic libraries to start testing with in the first place.
http://w3.ualg.pt/~jluis/

LAStools mailing list
http://groups.google.com/group/lastools

The work of authors has been partly supported from the following projects:

Academy of Finland Centre of Excellence in Laser Scanning Research (no. 272195), Academy of Finland projects "New techniques in active remote sensing: hyperspectral laser in environmental change detection" (no. 125447/218144) and "New Applications for Ubiquitous Multi- and Hyperspectral Mobile Mapping Systems" (no. 265949).

Function Overview of FGI matlas_tools

matlas_tools is a MATLAB mex gateway for using LasTools read and write functionalities from MATLAB.

matlas_tools consists of a reader function *las2mat*, a writer function *mat2las* and collection of helper functions and definitions (*mex_lasz_fun_fgi.cpp* and *mex_lasz_fun_fgi.hpp*).

matlas_tools copies the data fields from LasTools point and header objects into a MATLAB structure. The point-wise data are stored in MATLAB as a structure of vectors, not as in the LasTools, where each point is an object (the more precise MATLAB presentation would be a vector of structures). The structure of vectors in MATLAB is both faster and suits our purposes better.
There are 13 fields to the point data that are always copied / present and 3 fields that are present if the data field is in use in the LasTools point object. The persistent fields are present in each structure of points, even if they hold no data. They are:
"x", "y", "z", "intensity", "return_number", "number_of_returns", "scan_direction_flag", "edge_of_flight_line", "classification", "scan_angle_rank", "user_data and point_source_ID".
The fields "gps_time", "rgb" and "attributes" only appear if they are found in the data. The size of the fields "rgb" and "attributes" is column times row (m x n). The "rgb" field is of size N x 3, where N is the number of points.
This is important to note especially when constructing a structure in MATLAB for writing into a las file. The attributes appear both in the header and in the point data. In the header, field "attributes" is a substructure that has fields "name", "type", "description"," scale" and "offset" for each extra attribute. In the data, the values are saved in a column x row matrix in the same order they are listed in the header. The attribute values are already scaled using the parameters in the header.

Instructions for compiling matlas_tools in Linux

How to install LASlib and *matlas_tools*. These instructions have been tested with **Ubuntu 12.10** and **lastools v140322**.

1. extract the *matlas_tools* package, path denotes the path to *matlas_tools*
2. download the LASTools library (in the example below, it is extracted to the *matlas_tools* folder). If you have LASTools already, you still need to compile the shared library (edit the Makefile in */lastools/LASlib/src* directory to build .so) and fix the paths in the mex commands. Make sure your LASlib version is not older than 131025.
3. copy and rename the */path/matlas_tools/lastools/LASlib/src/Makefile* to */path/matlas_tools/lastools/LASlib/src/Makefile_orig*
4. edit the Makefile so that the *COPTS* line at the top has the "*-fPIC*" as the example Makefile in this package also edit lines all: *add liblas.so*.

NOTE: if you copy the Makefile from *matlas_tools* to *path/matlas_tools/lastools/LASlib/src/* it is important to check that the list of files (with .o ending) is the same as in Makefile_orig because there might be changes between the LASlib versions!

5. in terminal go to */path/matlas_tools/lastools/LASlib/src* and run *make* (type *make* and hit enter).
6. in Matlab go to directory */path/matlas_tools/* if you installed the lastools in the */path/matlas_tools/* folder do 7(a), otherwise do 7(b).
7. (a) type *compile_matlas_tools*, if there are no errors the reader is ready for use
7. (b) compile the *fgi_las2mat.cpp* and *fgi_mat2las* with the commands below:

NOTE! You need to replace '*path*' with the directory path that you are using!

```
mex las2mat.cpp /path/matlas_tools/lastools/LASlib/lib/liblas.a -
I/path/matlas_tools/lastools/LASlib/inc -I/path/matlas_tools/lastools/src -
I/path/matlas_tools/lastools/LASzip/inc -I/path/matlas_tools/lastools/LASzip/src
mex mat2las.cpp /path/matlas_tools/lastools/LASlib/lib/liblas.a -
I/path/matlas_tools/lastools/LASlib/inc -I/path/matlas_tools/lastools/src -
I/path/matlas_tools/lastools/LASzip/inc -I/path/matlas_tools/lastools/LASzip/src
```

8. To read a .las/.laz file, use the read command:

```
[hdr,str] = las2mat('-i /path/matlas_tools/lastools/data/house.laz');
```

And to write the structure back into .laz file, simple write:

mat2las(str,'-o house_copy.laz'); OR *mat2las(str,hdr,'-o house_copy.laz');*

For more examples, see the file ***Use_Example.m***.

Instructions for compiling matlas_tools in Windows

Preface

This is a short guide for compiling LAStools libraries with Microsoft Visual C++ 2010 Express. It has been tested with the test configuration listed below, but the guide comes ‘as is’ and with no guarantee of working.

Test configuration (as of March 2014):

Win 7, 64 bit
LAStools v140301
MATLAB 2012a

If you want to use precompiled binaries without compiling from the source, skip directly to part **6**.

1. Setting up Visual C++ 2010 Express for MATLAB

NOTE: Compiler Instructions for MATLAB **as of November 2012**, check for present compiler compatibilities from Mathworks webpage:
http://www.mathworks.se/support/compilers/R2012b/win64.html

Step 1:
Install NET 4.0 Framework from MSDN
(NET 4.5 or newer *may* also work, but these have not been tested)
(-> Reboot and run Windows update afterwards)
http://www.microsoft.com/en-us/download/details.aspx?id=17718 (64bit)

Step 2:
Install Windows SDK 7.1 from MSDN
(-> Reboot and run Windows Update afterwards)
http://www.microsoft.com/en-us/download/details.aspx?id=8279
(web-installer)

Step 3:
Install Visual C++ 2010 Express
[Reboot +] Update + Reboot
http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express

The number of reboots might be too excessive, but they are there to guarantee that register gets updated between installs.

*Update is **required** as earlier versions of the SDK and MSVC++ messed up the pathing for each other if they are installed in wrong order!*

NOTE: MS VC++ 2012 Express is also available, but the present (2012b) - nor the previous - MATLAB versions do NOT support it officially.

2. General compilation instructions in MS Visual C++ 2010 Express

NOTE!!
You have to configure each possible compilation setup explicitly, if you want to compile different versions. (Paths, compiler options, linker options etc.)
(32-b Debug / 32-b Release / 64-b Debug / 64-b Release)

Different compilation versions are selected from

- ➔ Configuration Manager (Upper right corner in the Project Property Window)
  - Change Platform into 'x64' instead of 'win 32'
    - Click open <'New...'>
    - Choose Release / Debug

NOTE!
These instructions are aimed for compiling the *whole* library. For particular functionalities and improved speed, only the required header and source files are needed.

TIP:
If a compilation fails to errors, build it nevertheless for the second time. The compiler will show only the errors on the second run making it easier to spot and fix them. Also, apply 'Clean Solution' after making changes to compiler options before trying to compile again.

3a. Compiling a static library from LASlib files

In Windows (copying performed for original file safety)
3a.0a Copy all *.hpp files from /LASlib/inc and /LASzip/src into the same folder
3a.0b Copy all *.cpp files from /LASlib/src and /LASzip/src into the same folder
3a.0c Copy also ‘geoprojection.cpp / .hpp’ under the ‘/src’ / ‘/inc’ folders

3a.1 Edit the header file 'mydefs.hpp' in line 59
FROM: #if defined(_MSC_VER) && (_MSC_VER < 1300)
TO: #if defined(_MSC_VER)

- This sets the definition of Boolean for MSVC compilers

OR

3a.Python Run the Python script provided in folder ‘/matlas_tools/OS_Win/Src2Lib’ with parameter ‘**1**’. The script creates new /src and /inc folders with edited source.

In Visual C++ 2010 Express
3a.2 Start a new project (class lib .lib)
- Delete all preset files from the project
- Add header and source files into the project ('Add existing item...')
- Add right folders from "Properties”
    - Configuration properties
        - VC++ Directories" to Include and Source
    - General
        - Platform Toolset (**Windows7.1SDK**)
        - Configuration Type(**Static library (.lib)**)
        - Target Extension(**.lib**)
        - Character Set(**Use Multi-Byte Character Set**)
        - Common Language Runtime Support (**NO** common …) [*optional?*]
    - C/C++
        - Precompiled Headers
            - Precompiler Header(**Not Using Precompiled Headers**)
        - Advanced
            - Compile conditions (**__cdecl (/Gd)**) [*optional?*]
            - Compile As(**Compile as C++ Code (/TP)**) [*optional?*]

3a.3 Cross your fingers and hit 'Build Solution'. There should be no errors, but lots of warnings that can be ignored.

3b. Compiling a dynamic library from LASlib files

In Windows (copying performed for original file safety)
3b.0a Copy all *.hpp files from /LASlib/inc and /LASzip/src into the same folder
3b.0b Copy all *.cpp files from /LASlib/src and /LASzip/src into the same folder
3b.0c Copy also ‘geoprojection.cpp / .hpp’ under the /src

In Windows, required for DLL
3b.1a Copy 'export_symb.h' into the header folder (provided by Joaquim Luis, the file basically changes class definitions so that they work in DLL creation)
3b.1b Add #include "export_symb.h" in ...
'lasdefinitions.hpp'
'mydefs.hpp'
‘laszip.hpp’
‘geoprojectionconverter.hpp’
(check for others: Look for errors about ‘CPL_DLL’ during the building)

3b.2 Remove ‘LASZIP_DLL’ in the front of class definitions in all header and source files, **if** present. The definition seems to be empty at present. (At least in ‘laszip.hpp’)

3b.3 In **every included** header (and source) file that contains class definitions, add ‘CPL_DLL’ in front of each class (CPL_DLL is defined in export_symb.h).
e.g. FROM: class LAScriterion{…
TO: class CPL_DLL LAScriterion{…

3b.4 Edit the header file 'mydefs.hpp' in line 59
FROM: #if defined(_MSC_VER) && (_MSC_VER < 1300)
TO: #if defined(_MSC_VER)

- This sets the definition of Boolean for MSVC compilers

OR

3b.Python Run the Python script provided in folder ‘/matlas_tools/OS_Win/Src2Lib’ with parameter ‘**0**’. The script creates new /src and /inc folders with edited source.

In Visual C++ 2010 Express
3b.5 Start a new project (class lib .lib) **[!]**
- Delete all preset files from the project
- Add header and source files ('Add existing item...')
- Add right folders from "Properties”
    - Configuration properties
        - VC++ Directories" to Include and Source

- General
    - Platform Toolset (**Windows7.1SDK**)
    - Configuration Type(**Dynamic Linking Library (.dll)**)
    - Target Extension(**.dll**)

    - Character Set(**Use Multi-Byte Character Set**)
    - Common Language Runtime Support (**NO** common …) [*optional?*]
- C/C++
    - Precompiled Headers
        - Precompiler Header(**Not Using Precompiled Headers**)
- Advanced
    - Compile conditions (**__cdecl (/Gd)**) [*optional?*]
    - Compile As(**Compile as C++ Code (/TP)**) [*optional?*]

3b.6. Cross your fingers and hit 'Build Solution'. There should be no errors, but lots of warnings that can be ignored.

4. Test compilation with LAStools LGPL source files

In Visual C++ 2010 Express
4.1 Start a new project (Win32 Console Application)
• In the opening window, remove 'Preset headers' tick from 'Application settings'
• Delete all preset files from the project
• *Add only the .cpp source file that you want to compile (*'Add existing item...*') to the* 'Source files' *and leave other folders empty* (library will provide the rest)

4.2 Set compiler options, use the same options as before depending on the selected library (.lib / .dll)
BUT
•Check under "Properties" that
- •General
  - •Configuration Type(**Application (.exe)**)
  - •Target Extension(**.exe**)

4.3 Compiler pathing
•Add right folders from "Properties"
- •Configuration properties
  - "VC++ Directories"
    - •For include add the same header folders as with the library
    - •For source add the source file folder
    - •For library add the folder of the precompiled library
      NOTE: Check that the file version (release/debug) and platform type (32-b/64-b) match!

4.4 Linker options
•Add the correct library file for the linker
- Linker -> Input -> Additional dependencies -> Library file name
  NOTE! When compiling the .dll version, one still has to link against a .lib file generated at the same time with the .dll library. The .lib file is located in the same folder as the .dll file.

4.5 Cross your fingers and hit 'Build Solution'. There should be no errors, but maybe a few warnings.

4.6. Start the command line in windows and try the new binary as usual.
NOTE! If you compiled the binary using the .dll library, then you have to have the .dll file in the same folder with the .exe to make it run (or edit path environment).

5. Using compiled libraries in MATLAB with mex

IF library compilation was successful, then test with a 'real' mex code.

5.0 If mex is not setup yet, write 'mex -setup' and MATLAB should recognize the Microsoft compiler.

5.1 Compile your code with :

mex [-O] [-largeArrayDims] -I[Path\to\Headers\] MyCode.cpp -L[Path\to\Libs\] -l[LibFile]
e.g.

tic, mex las2mat.cpp -O -largeArrayDims -I\PathToLib\Def_LIBS\LIBinc -
I\PathToLib\Def_LIBS\ZIPsrc -L\PathToLib\ -lStatLib64b; toc

tic, mex mat2las.cpp -O -largeArrayDims -I\PathToLib\Def_LIBS\LIBinc -
I\PathToLib\Def_LIBS\ZIPsrc -L\PathToLib\ -lStatLib64b; toc

OR

Run **Make_binaries.m** after extracting the “Win” folder toolbox.

6. Simple use examples with matlas_tools binaries

To read a .las/.laz file, use the read command:

[hdr,str] = las2mat('-i /path/matlas_tools/lastools/data/house.laz');

And to write the structure back into another .laz file, simply write:

mat2las(str,'-o house_copy.laz'); OR *mat2las(str,hdr,'-o house_copy.laz');*

For more examples, see file **Use_Example.m**.

NOTE! If you have compiled your mex binary with a dll library, then the dll file has to be in the same folder (or path) with the mex-file to run! Linking with a static library results in a stand-alone mex-file.

7. FAQ

Q. Why are .las data saved into a single structure with vectors and matrices in the MATLAB workspace (structure of arrays) instead of saving each point as their own native structure (array of structures)?
A. We tested saving data with the (native) array of structures format, but this implementation was i) several times slower and ii) required several times more memory in workspace than the present implementation. However, this decision requires a user to keep careful account of matrix sizes in all variables.

Q. Why is every value read as a double into the MATLAB workspace?
A. MATLAB uses double values natively. Also, double values have high enough precision to guarantee that long numeral values, e.g. coordinates in some reference frames, won't be clipped.

Q. Lasreader/laswriter option "X" is not working?
A. Our main aim was to provide a .mex gateway for accessing and writing .las files in MATLAB and not to implement all possible LASlib functionalities. (Properties of) the other lastools functions can be accessed within MATLAB with *system* command.

Q. Library and static binary file sizes in the matlas package are multiple times of that in the original lastools package. What is the reason for this?
A. The most likely reason is that MATLAB requires certain dependencies for compiling .mex, e.g. .NET framework ver 4.1 and Windows SDK ver 7.1. These are likely to add to the size of the final files. Also, we are including 'geoprojection.hpp/cpp' in our library files that can be dropped out. The [optional] use of compiler option '\clr, common language runtime support' will add to the final library size.

Q. Will your matlas code also compile for Octave .mex files? And what measures would be required for that?
A. This has not been tested, but we would be most interested to hear if someone with Octave experience would try to make an implementation.