McCAD v1.0
User Manual

Moataz Harb
moataz.harb@kit.edu
Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen, Germany

May 24, 2022

Contents

0.1 Introduction

McCAD is a C++ library for the conversion of CAD solid models to MCNP input syntax, from Boundary Representation "BREP" to Constructive Solid Geometry "CSG".

0.2 Installation from Source

Building of the McCAD library is supported on both Linux and Windows operating systems. The library has three 3rd-party dependencies: CMake, Boost C++ libraries, and Open CASCADE Technology. CMake is the standard build system for McCAD and it comes by default with most Linux distributions as well as Windows OS. Boost C++ libraries consist of header files that are utilized for parallel processing in McCAD. Open CASCADE Technology (OCCT) is used as a geometry engine for geometrical solids manipulation and decomposition. Below are guiding steps for installation on both Linux and Win systems.

0.2.1 Linux

Listed below are the currently supported Linux distributions:

- Ubuntu 20.04 LTS
- Ubuntu 18.04 LTS

Testing of installation on other distributions is still underway! thus Ubuntu is recommended as a distribution to install McCAD. Below are general steps to install McCAD code and its dependencies.

- **CMake**
  - Download cmake-3.23.0.tar.gz from https://cmake.org/download/ then execute the commands below in a terminal.
  - $ tar -xzvf cmake-3.23.0.tar.gz
  - $ cd cmake-3.23.0
  - $ mkdir build
  - $ cd build
  - $ cmake .. -DCMAKE_USE_OPENSSL=OFF -DCMAKE_INSTALL_PREFIX=.
  - $ make
  - $ make install

- **Boost C++ liraries**
  - Download boost_1_78_0.tar.gz from https://www.boost.org/users/download/ then execute the commands below in a terminal.
  - $ tar -xvzf boost_1_78_0.tar.gz
  - $ cd boost_1_78_0
  - $ mkdir build
  - $ cd tools/build
  - $ ./bootstrap.sh
  - $ ./b2 install --prefix=../../build/
- **Open CASCADE Technology (OCCT)**
  - *NOTE*: the instructions on the installation of dependencies can be found on the side menu in https://dev.opencascade.org/doc/occt-7.5.0/overview/html/index.html by navigating to "Build, Debug and Upgrade > Build 3rd-parties" then following the instructions under "Installation from Official Repositories".
  - Download opencascade-7.5.0.tgz from https://dev.opencascade.org/release/previous then execute the commands below in a terminal.
  - $ tar -xzvf opencascade-7.5.0.tgz
  - $ cd opencascade-7.5.0
  - $ mkdir build
  - $ cd build
  - $ cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_LIBRARY_TYPE=Shared -DCMAKE_INSTALL_PREFIX=. -DINSTALL_TEST_CASES=TRUE -DINSTALL_DOC_Overview=TRUE
  - $ make
  - $ make install
- **McCAD**
  - *NOTE*: building a shared library is recommended! Should a static library be needed, the user has to insure a compliant build of Open CASCADE Technology by changing the build type; -DBUILD_LIBRARY_TYPE=STATIC.
  - $ git clone https://github.com/inr-kit/McCAD_Library
  - $ cd McCAD_Library

- $ mkdir build
- $ cd build
- $ CMake .. -DCMAKE_INSTALL_PREFIX=. -DBUILD_STATIC=OFF -DBOOST_CUSTOM_ROOT=<PATH to boost_1_78_0> -DOCC_CUSTOM_ROOT=<P to opencascade-7.5.0/build> -DBUILD_RPATH=ON
- $ make
- $ make install

0.2.2 Windows

Listed below are the currently supported Windows versions:

- Windows 10

Testing of installation on other versions is still underway! Below are general steps to build McCAD library and its dependencies.

- **CMake** (optional)
  - *NOTE*: If usage of IDE - such Microsoft Visual Studio (VS) - is intended, then installing CMake can be skipped since most IDE builds CMake by default.
  - Download and run the installer cmake-3.23.1-windows-x86_64.msi from https://cmake.org/download/.
- **Microsoft Visual Studio** (optional)
  - Download and run the "community" installer from https://visualstudio.microsoft.com/downloads/.
- **Boost C++ liraries**
  - Download boost_1_78_0.zip from https://www.boost.org/users/download/.
  - Unzip boost_1_78_0.zip.
  - Documentation can be found in index.html in the unzipped folder.
- **Open CASCADE Technology (OCCT)**
  - Download and run the installer opencascade-7.5.0-vc14-64.exe from https://dev.opencascade.org/release/previous.
- **McCAD**

- Download source code from https://github.com/inr-kit/McCAD_Library by selecting Code > Download ZIP.
- Unzip McCAD_Library.
- Open MSVC and select the McCAD_Library.
- From the "Solution Explorer - Folder Review" double click CMakeSettings.json file. This will open the file in IDE.
- Set a "Configuration name".
- Ensure that "Configuration type" is set to "Release" and "Toolset" is set to msvc_x64_x64.
- Add -DBUILD_RPATH=ON -DBUILD_STATIC_EXE=ON -DBUILD_SHARED=OFF -DBOOST_CUSTOM_ROOT="<PATH to boost_1_78_0>" -DOCC_CUSTOM_ROOT="<PATH to OpenCASCADE-7.5.0-vc14-64\opencascade-7.5.0>" to "CMake command arguments".
- From the top menu select "Build > Build All".
- From the top menu select "Build > Install McCAD".

0.3 I/O

In general, there are three types of information that need to be provided to McCAD: geometry model(s), materials assignment, and running parameters. Below, the input pertinent to the decomposition and conversion will be highlighted. The output from McCAD will depend on the mode of execution, as will be described in the following sections.

0.3.1 General

The first input to be provided is the input configuration file, McCADInputConfig.i. When run via a command line *without arguments*, McCAD will generate a template configuration file, figure (1), with the default values of the run parameters. In both run modes, decomposition and conversion, McCAD input parameters are needed. The run parameters are assigned values by default and can be modified by the user. Below is a description of the general input parameters.

- **debugLevel**: the level at which run/debug info is diplayed on the screen. *Default* is 0.
- **units**: The units used in the CAD model. *Default* is cm.
- **inputFileName**: the input STEP file(s) name(s). *Default* is input.stp.

```
# McCAD Run Parameters    Mon May 23 17:10:46 2022
# ====================

# Input
# =====
# > Debug level: 0, 1, 2, 3. [0] provides no debugging outputs.
debugLevel = 0
# > The unit used for the input STEP file(s).
units = cm
# > Path to the input STEP file;
inputFileName = input.stp

# Decomposition
# =============
decompose = true
recurrenceDepth = 20
minSolidVolume = 1.0e-3 [cm3]
minFaceArea = 1.0e-4 [cm2]
scalingFactor = 100.0
precision = 1.0e-6
faceTolerance = 1.0e-8 [cm]
edgeTolerance = 1.0e-8 [cm]
parameterTolerance = 1.0e-8 [cm]
angularTolerance = 1.0e-4 [radian/PI]
distanceTolerance = 1.0e-6 [cm]
simplifyTori = false
simplifyAllTori = false
torusSplitAngle = 30.0 [degrees]

# Conversion
# ==========
convert = false
# > Choose whether or not to generate void cells;
voidGeneration = false
# > Condition to treat a compound as a single cell or a group of cells.
compoundIsSingleCell = false
# > Minimum acceptable void volume shouldn;t be less than minSolidVolume;
minVoidVolume = 1.0 [cm3]
# > A larger number will result in fewer void cells but longer cell expressions;
maxSolidsPerVoidCell = 40
# > Choose whether or not to generate Bound Volume Heirarchy void cells;
BVHVoid = false
# > Choose the desired MC code for conversion;
MCcode = mcnp
startCellNum = 1
startSurfNum = 1
startMatNum = 1
maxLineWidth = 80
MCFileName = MCFile.i
volumesFileName = volumes.i
voidCellsFileName = voidCells.i
```

Figure 1: McCAD Input Configuration File

0.3.2 Decomposition

Currently McCAD supports input files in the STEP format (protocol 214). McCAD accepts both single and multiple input STEP files. After successfully running McCAD decomposition two STEP output files will be written to disk. The first will contain the decomposed input solids and the second will contain the rejected/failed input solids. If the latter is absent, this means no input solids were rejected and no solids failed to be decomposed.

Before running McCAD in the decomposition mode, the user is advised to review the input parameters on the decomposition section on the input configuration file, figure (1). Below is a description of the decomposition input parameters.

- **decompose**: a logical condition for running decomposition on the input file(s). *Default* is true.
- **recurrenceDepth**: defines the level of decomposition performed recursively on input solids. This input specifies the maximum number of recursive application of decomposition on the subsolids. The larger the number, the less the failed solids with a lot of detailes in the geometry. *Default* is 20.
- **minSolidVolume**: this input is used to control the recursive splitting of the solids along with the reciurrenceDepth. It is also used to filter the input solids by rejecting smaller solids than the specified value. *Default* is 1.0e-3 cm^3.
- **minFaceArea**: used to filter the input solids by rejecting solids with faces smaller in area than the specified value. *Default* is 1.0e-4 cm^2.
- **scalingFactor**: controls the size of generated mesh on solids faces. This parameter mainly affects the collision detection of surfaces. More details can be found in section 0.7. *Default* is 100.0.
- **precision**: used to judge the equality of numerical values. *Default* is 1.0e-6.
- **faceTolerance**: More details can be found in section 0.7. *Default* is 1.0e-8 cm.
- **edgeTolerance**: More details can be found in section 0.7. *Default* is 1.0e-8 cm.
- **parameterTolerance**: More details can be found in section 0.7. *Default* is 1.0e-8 cm.

- **angularTolerance**: More details can be found in section 0.7. *Default* is 1.0e-4 radian/PI.
- **distanceTolerance**: More details can be found in section 0.7. *Default* is 1.0e-6 cm.
- **simplifyTori**: a logical condition on whether or not simplify toroidal solids as a collection of smaller cylindrical segments. If true, only off axis tori will be simplified. *Default* is false.
- **simplifyAllTori**: a logical condition on whether or not simplify all toroidal solids as a collection of smaller cylindrical segments. If true, all toroidal solids, disregarding its axis, will be simplified. *Default* is false.
- **torusSplitAngle**: the angle between the top and bottom bases of a cylindrical segment. *Default* is 30.0 degrees.

0.3.3 Conversion

McCAD conversion will result in three textual output files. The first contains the generated Monte Carlo, MCNP is only support for now, input file. The second contains a mapping of cell ID, volume, and solid name. The third file contains a mapping of void cell IDs to material cell IDs. More details on the format of the three files can be found in section 0.5.

Material assignment is currently deduced from the name of the STEP files. The user is advised to split the CAD model into several by material. The name of the input STEP files takes the format <material name>_<density in exponential format: -0.0E0>_<.stp. The material name and density are then extracted from the file name and assigned to the solids it contains. For MCNP, the sign of the density implies whether mass or atomic densities is used. In case of absence of the density value, void is assumed. It is important to note that the underscore only separates the material name from density. Any extra underscores will result in an error.

Before running McCAD in the conversion mode, the user is advised to review the input parameters on the conversion section on the input configuration file, figure (1). Below is a description of the decomposition input parameters.

- **convert**: a logical condition for running conversion on the input file(s). *Default* is false.

- **voidGeneration**: a logical condition for void generation. If false, only a single void cell is created outside which is a graveyard. *Default* is false.
- **compoundIsSingleCell**: controls whether to define the subsolids of the decomposid input solid into separate cells or unify it through the union operator into a single cell. If true, all subsolids are joined through a union operator and a single cell is written to the MC input file. This is not recommended for complex solids since it slows down the transport due to the many surface crossing checks. *Default* is false.
- **minVoidVolume**: controls the desirable minimum void cell volume. *Default* is 1.0 cm^3.
- **maxSolidsPerVoidCell**: this input will control the number of generated void cells. A smaller number will result in a larger number of void cells each with simpler expression on the MC file. *Default* is 40.
- **BVHVoid**: a logical condition for generating a bound volume hierarchy void cells. This feature is still undergoing testing and not currently recommended to be used. *Default* is false.
- **MCcode**: specifies the targeted MC code to generate the input file in a compliant format. Currently only MCNP is supported. *Default* is mcnp.
- **startCellNum**: the desired starting cell number. *Default* is 1.
- **startSurfNum**: the desired starting surface number. *Default* is 1.
- **startMatNum**: the desired starting material number. *Default* is 1.
- **maxLineWidth**: maximum length of lines on the MC file. *Default* is 80.
- **MCFileName**: the desired name for the MC file. *Default* is MCFile.i.
- **volumesFileName**: the desired name for the cell ID-volumes-name mapping file. *Default* is volumes.i.
- **voidCellsFileName**: the desired name of the void to material cells mapping file. *Default* is voidCells.i.

0.4 Running McCAD

There are two ways to run McCAD, via a command line or the SpaceClaim plugin. McCAD can be run from a command line either in a Linux terminal or a Windows command prompt.

0.4.1 Command Line

On a linux termial of Windows command prompt McCAD can be run by typing in the path to the executable followed by the desired argument. A good practice is to add the path to the McCAD executable to the PATH environment variable then simply call McCAD with the acceptable desired command line arguments. The arguments could be displayed on the screen by running "McCAD help". The acceptable McCAD arguments are listed below.

- []: running McCAD without arguments will result in writing McCAD template input configuration file, McCADInputConfig.i.
- [**help**]: lists all the acceptable arguments.
- [**read**]: test loading the solids from the input STEP file(s). This is used to test if the OCCT reader will fail to load any STEP file(s).
- [**run**]: executes McCAD according to the selected running mode: decomposition, conversion, or both.

0.4.2 SpaceClaim Plugin

The python script located in SpaceClaim_interface can be loaded into SpaceClaim and used directly to call McCAD to perform decomposition on selected solid(s). The script can be loaded in SPaceClaim by selecting "File > New > Script". Select the desired solid(s) to decompose and on the "Script Editor" select "Run Script", the green arrow.

0.5 Example

General notes on how to use McCAD. Coming soon ...

0.5.1 Decomposition

0.5.2 Conversion

- Coordinates

0.6 Known Issues

A list of known issues and how to manually fix it in solid models. Coming soon ...

0.6.1 Decomposition

- simplifying of tori
- hollow solids
- splitting of cylinders and tori

0.6.2 Conversion

0.7 Theory of McCAD Conversion

Comperhensive details on the inner workings of McCAD classes for developers. Coming soon ...

0.7.1 Decomposition

0.7.2 Conversion