

Plugin Creation with Existing MITK project

Zaiwang Zhou

December 6, 2021

Contents

1	Preparation	1
1.1	Overview of an MITK plugin	1
1.2	Enabling of PointSetInteraction	2
2	Creation of an MITK module	3
2.1	Setup of the directories	3
2.1.1	autoload	3
2.1.2	cmdapps	4
2.1.3	doc	4
2.1.4	include	4
2.1.5	resource	4
2.1.6	src	5
2.1.7	test	5
2.2	Transfer of the module code	5
3	Creation of an MITK plugin	6
4	Demonstration of the newly created plugin	8

1 Preparation

1.1 Overview of an MITK plugin

A plugin provides extended functionality of MITK by utilizing the functions encapsulated inside the MITK modules. In *MITK Compilation Guide for Windows 10 machines*, the compilation of an MITK project which was modified by our company has been introduced. The GUIs of the resulted MITK workbench is shown in Figure 1.1. The panel on the right named "VC view" is a plugin which can be found under "Window" → "Show view". What we hope to achieve with this manual is to add another new "View" to this existing workbench. For demonstration purposes, the source code of the new plugin will be provided in the folder "files_needed". The new plugin is developed for image registration with ICP (iterative closest point).

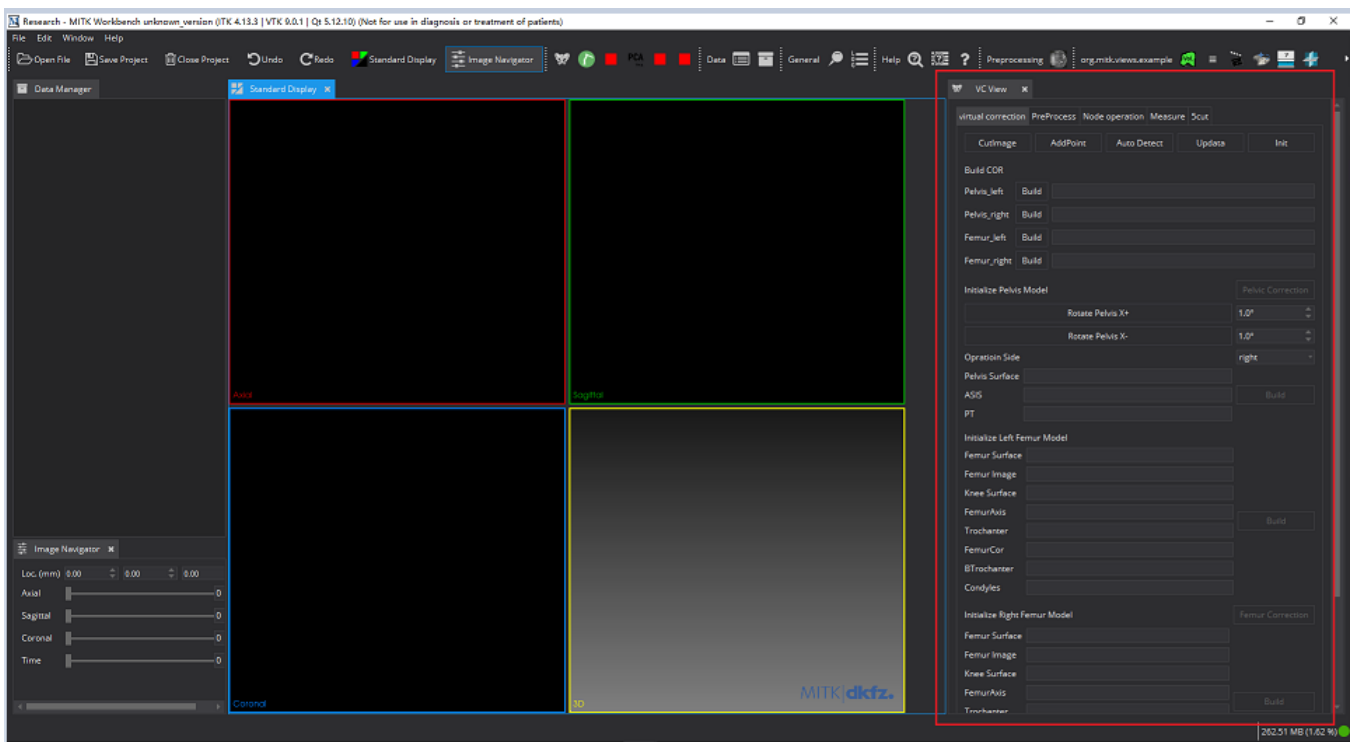


Figure 1.1: GUIs of the MITK workbench with an "VC View" plugin

1.2 Enabling of PointSetInteraction

In order to let users interactively set points on the surface mesh with the MITK workbench, the `PointSetInteraction` plugin should be enabled with CMake. The directories and the complete plugin name are shown in Figure 1.2. After "Configure", "Generate" and "build (in VS)", run the MITK workbench and you can find "Point set interaction" under "Window" → "Show view".

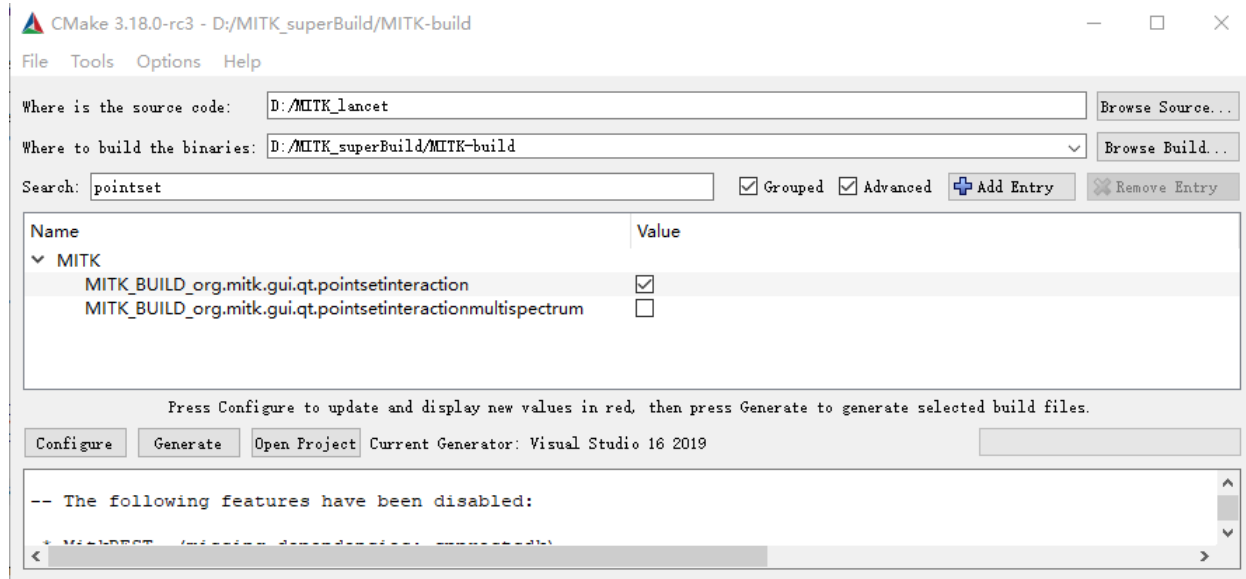


Figure 1.2: Enabling the PointSetInteraction module with CMake

2 Creation of an MITK module

2.1 Setup of the directories

First of all, a folder named "SurfaceRegistration" is created under the "Modules" folder within your local MITK repository:

```
1 D:\MITK_lancet\Modules\
```

Then open `D : \MITK_lancet\Modules\ModuleList.cmake` and append the folder to the MITK_MODULES list as below. Notably, the sequence of the modules is crucial. If module B depends on module A, B should be placed after A!

```
1 set(MITK_MODULES
2     ...
3     SurfaceRegistration
4 )
```

A standard MITK module named "ModuleA" should have a directory structure as in Figure 2.1. Depending on your use case you might not need every directory (as we really did for SurfaceRegistration module). A quick description of what each directory contains is provided. This information is just for your reference and not so important, since the source code of the new plugin is already given. What you need to do is just to place the files in the correct location as described in the Section 2.2.

2.1.1 autoload

This directory should not directly contain files. Instead it contains further directories each of which is its own module. These modules provide functionality which should be available at application start, but will not be included by other modules. Examples would be a module registering readers/writers or providing an interface for specific hardware. For an example of an autoload module take a look at NewModule/autoload-

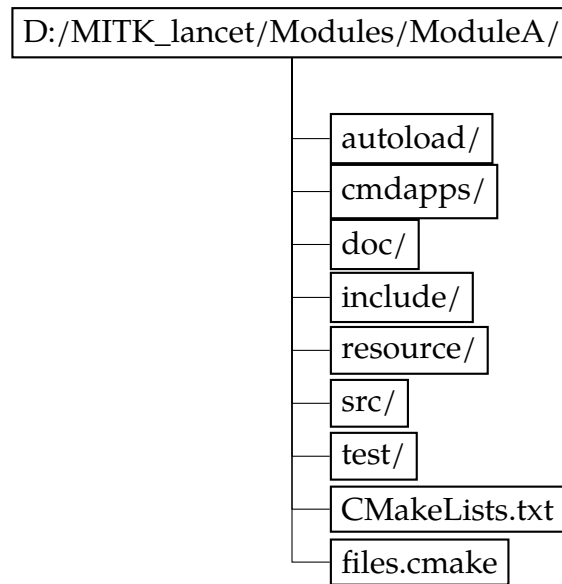


Figure 2.1: Directory structure of a typical MITK module

/IO.

2.1.2 cmdapps

This directory contains all cmdapps (command line tools) related to your module.

2.1.3 doc

This directory contains the documentation for your module.

2.1.4 include

This directory contains all header files which might be included by other modules.

2.1.5 resource

This directory contains resources needed by your module, such as xmls, images or similar.

2.1.6 src

This directory contains source and header files which should not be included by other modules. Further subdivision can help keeping it organized. (e.g. src/DataManagement src/Algorithms)

2.1.7 test

This directory contains all tests for your module.

2.2 Transfer of the module code

In short, what you need to do is just to move the content of *files_needed\SurfaceRegistration* to your repository *D : \MITK_{lancet}\Modules*. There are some reminders though, when you compose your own module based on existing ones in the future:

- Properly edit "CMakeLists.txt" and "files.cmake" to fit your directory structure
- Use the proper macros in the header file for DLL building. ("MITKSURFACE-REGISTRATION_EXPORT" in our example)



ITK&VTK modules are added here!

3 Creation of an MITK plugin

The MITK Plugin Generator is a command line tool to simplify the process of creating your own MITK plugins. Please use it only with cmd in Windows, since it seems to have some incompatibility with Powershell. The Plugin Generator takes the following command line arguments:

```
1 ./MitkPluginGenerator -h
2 A CTK plugin generator for MITK (version 1.2.0)
3
4 -h, --help                Show this help text
5 -o, --out-dir             Output directory (default: /tmp)
6 -l, --license             Path to a file containing license
                           information (default: ./COPYRIGHT_HEADER)
7 -v, --vendor             The vendor of the generated code (
                           default: German Cancer Research Center (DKFZ))
8 -q, --quiet              Do not print additional
                           information
9 -y, --confirm-all        Answer all questions with 'yes'
10 -u, --check-update       Check for updates and exit
11 -n, --no-networking      Disable all network requests
12
13 Plugin View options
14 -vc, --view-class        The View's' class name
15 -vn, --view-name         * The View's human readable name
16
17 Plugin options
18 -ps, --plugin-symbolic-name * The plugin's symbolic name
19 -pn, --plugin-name       The plugin's human readable name
20
21 Project options
22 --project-copyright      Path to a file containing
                           copyright information (default: ./LICENSE)
23 --project-name           The project name
24 --project-app-name       The application name
25
26 [* - options are required]
```


Before using the MITK Plugin Generator, enter the directory of its executable:

```
1 D:\MITK_superBuild\MITK-build\bin\Release
```

We then create a plugin as Figure 3.1. To avoid errors, the "--plugin-symbolic-name" should start with "org.mitk.lancet".

```
D:\MITK_superBuild\MITK-build\bin\Release>. \MitkPluginGenerator --plugin-symbolic-name org.mitk.lancet.nodeeditor --view-name "Node Editor" --out-dir D:\MITK_lancet\Plugins
Using the following information to create a plug-in:

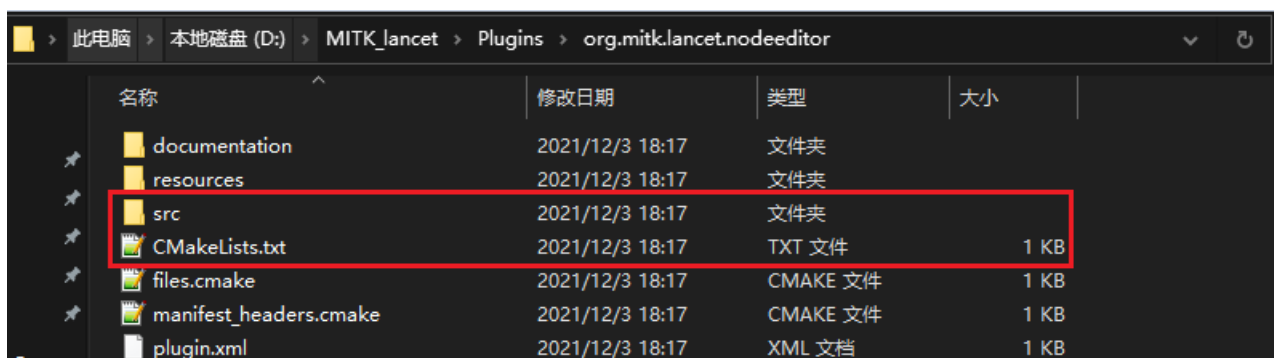
License File:      :\COPYRIGHT_HEADER
Plugin-SymbolicName: org.mitk.lancet.nodeeditor
Plugin-Name:       Nodeeditor
Plugin-Vendor:     German Cancer Research Center (DKFZ)
View Name:         Node Editor
View Id:           org.mitk.views.nodeeditor
View Class:        NodeEditor

Create in: D:/MITK_lancet/Plugins/org.mitk.lancet.nodeeditor

Continue [Y/n]? y
Directory 'D:/MITK_lancet/Plugins/org.mitk.lancet.nodeeditor' does not exist. Create it [Y/n]? y
```

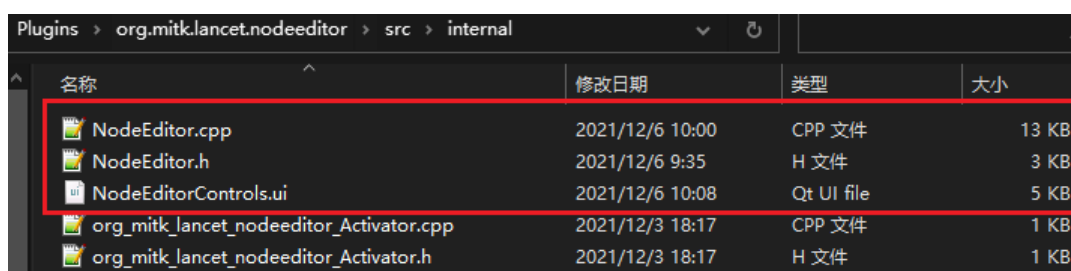
Figure 3.1: Generation of a new MITK plugin

The following directory structure (Figure 3.2) is generated automatically. For this manual, only the ones within the red box need to be modified. To include the needed modules into our new plugin, replace "CMakeLists.txt" with the one provided in the folder "files_needed". Then replace the files (Figure 3.3) in `\src\internal\` with the corresponding ones inside "files_needed".



名称	修改日期	类型	大小
documentation	2021/12/3 18:17	文件夹	
resources	2021/12/3 18:17	文件夹	
src	2021/12/3 18:17	文件夹	
CMakeLists.txt	2021/12/3 18:17	TXT 文件	1 KB
files.cmake	2021/12/3 18:17	CMAKE 文件	1 KB
manifest_headers.cmake	2021/12/3 18:17	CMAKE 文件	1 KB
plugin.xml	2021/12/3 18:17	XML 文档	1 KB

Figure 3.2: Automatically generated files and folders



名称	修改日期	类型	大小
NodeEditor.cpp	2021/12/6 10:00	CPP 文件	13 KB
NodeEditor.h	2021/12/6 9:35	H 文件	3 KB
NodeEditorControls.ui	2021/12/6 10:08	Qt UI file	5 KB
org_mitk_lancet_nodeeditor_Activator.cpp	2021/12/3 18:17	CPP 文件	1 KB
org_mitk_lancet_nodeeditor_Activator.h	2021/12/3 18:17	H 文件	1 KB

Figure 3.3: Source code of the new plugin

4 Demonstration of the newly created plugin

Use CMake to "Configure" and "Generate" again before going to VS. If everything is correct, after building the MITK Workbench, you will find "Node Editor" under "Window" → "Show view".

The test data "landmark.mtk" for the new plugin can be found in "files_needed". After loading the data, you may find the Workbench as in Figure 4.1. The newly added plugin is marked with a red box.

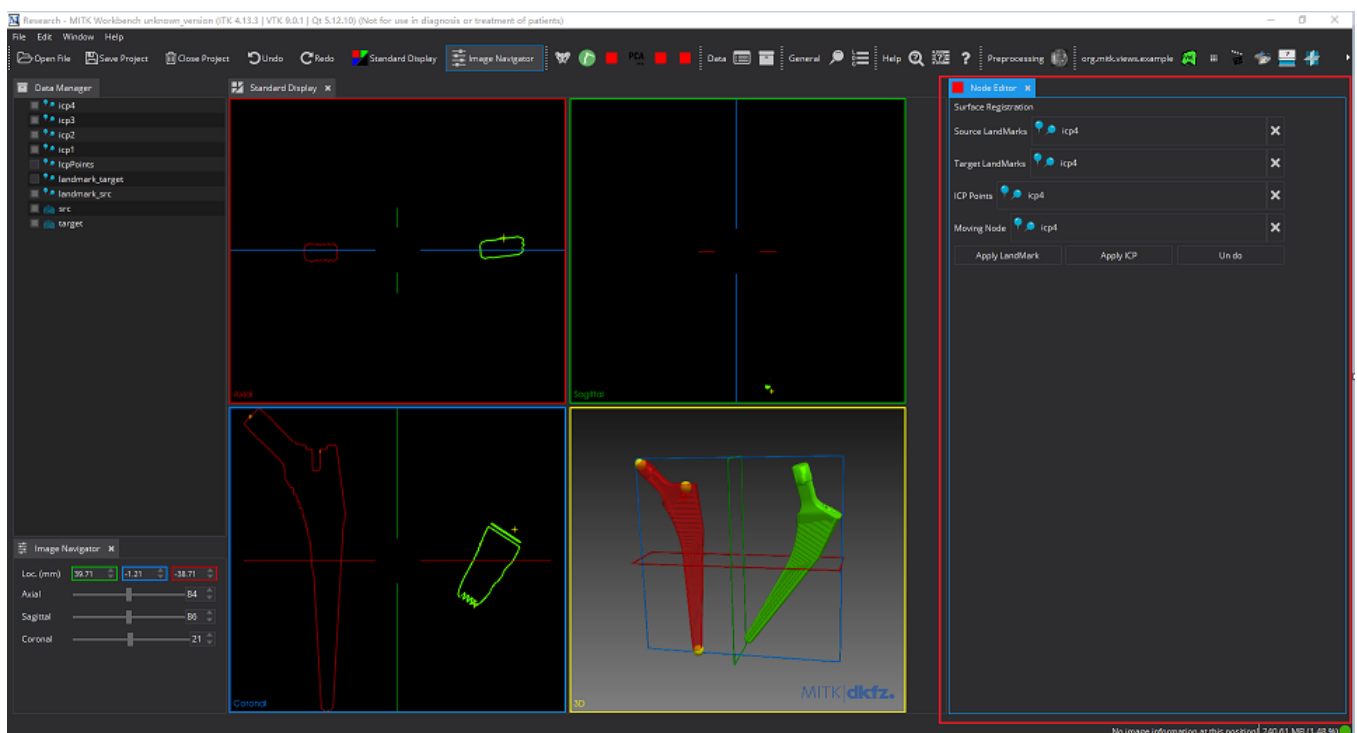


Figure 4.1: The MITK Workbench after loading the test data