cmake\_minimum\_required(VERSION 3.5.1)

project(grid\_map\_core)

set(CMAKE\_CXX\_FLAGS "-std=c++11 ${CMAKE\_CXX\_FLAGS}")

add\_compile\_options(-Wall -Wextra -Wpedantic)

set(CMAKE\_EXPORT\_COMPILE\_COMMANDS ON)

## Find catkin macros and libraries

find\_package(catkin REQUIRED)

## Define Eigen addons.

include(cmake/${PROJECT\_NAME}-extras.cmake)

## System dependencies are found with CMake's conventions

#find\_package(Eigen3 REQUIRED)

# Solution to find Eigen3 with Saucy.

find\_package(Eigen3 QUIET)

if(NOT EIGEN3\_FOUND)

find\_package(PkgConfig REQUIRED)

pkg\_check\_modules(EIGEN3 REQUIRED eigen3)

set(EIGEN3\_INCLUDE\_DIR ${EIGEN3\_INCLUDE\_DIRS})

endif()

###################################

## catkin specific configuration ##

###################################

## The catkin\_package macro generates cmake config files for your package

## Declare things to be passed to dependent projects

## INCLUDE\_DIRS: uncomment this if you package contains header files

## LIBRARIES: libraries you create in this project that dependent projects also need

## CATKIN\_DEPENDS: catkin\_packages dependent projects also need

## DEPENDS: system dependencies of this project that dependent projects also need

catkin\_package(

INCLUDE\_DIRS

include

${EIGEN3\_INCLUDE\_DIR}

LIBRARIES

${PROJECT\_NAME}

CATKIN\_DEPENDS

DEPENDS

#Eigen3

CFG\_EXTRAS

${PROJECT\_NAME}-extras.cmake

)

###########

## Build ##

###########

## Specify additional locations of header files

include\_directories(

include

SYSTEM

${catkin\_INCLUDE\_DIRS}

${EIGEN3\_INCLUDE\_DIR}

)

## Declare a cpp library

add\_library(${PROJECT\_NAME}

src/GridMap.cpp

src/GridMapMath.cpp

src/SubmapGeometry.cpp

src/BufferRegion.cpp

src/Polygon.cpp

src/CubicInterpolation.cpp

src/iterators/GridMapIterator.cpp

src/iterators/SubmapIterator.cpp

src/iterators/CircleIterator.cpp

src/iterators/EllipseIterator.cpp

src/iterators/SpiralIterator.cpp

src/iterators/PolygonIterator.cpp

src/iterators/LineIterator.cpp

src/iterators/SlidingWindowIterator.cpp

)

target\_link\_libraries(${PROJECT\_NAME}

${catkin\_LIBRARIES}

)

#############

## Install ##

#############

# Mark executables and/or libraries for installation

install(

TARGETS ${PROJECT\_NAME}

ARCHIVE DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

LIBRARY DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

RUNTIME DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

)

# Mark cpp header files for installation

install(

DIRECTORY include/${PROJECT\_NAME}/

DESTINATION ${CATKIN\_PACKAGE\_INCLUDE\_DESTINATION}

FILES\_MATCHING PATTERN "\*.hpp"

)

# Mark other files for installation

install(

DIRECTORY doc

DESTINATION ${CATKIN\_PACKAGE\_SHARE\_DESTINATION}

)

#############

## Testing ##

#############

if(CATKIN\_ENABLE\_TESTING)

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -pthread")

## Add gtest based cpp test target and link libraries

catkin\_add\_gtest(${PROJECT\_NAME}-test

test/test\_grid\_map\_core.cpp

test/test\_helpers.cpp

test/CubicConvolutionInterpolationTest.cpp

test/CubicInterpolationTest.cpp

test/GridMapMathTest.cpp

test/GridMapTest.cpp

test/GridMapIteratorTest.cpp

test/LineIteratorTest.cpp

test/EllipseIteratorTest.cpp

test/SubmapIteratorTest.cpp

test/PolygonIteratorTest.cpp

test/PolygonTest.cpp

test/EigenPluginsTest.cpp

test/SpiralIteratorTest.cpp

test/SlidingWindowIteratorTest.cpp

)

target\_include\_directories(${PROJECT\_NAME}-test PRIVATE

include

)

target\_include\_directories(${PROJECT\_NAME}-test SYSTEM PUBLIC

${catkin\_INCLUDE\_DIRS}

${EIGEN3\_INCLUDE\_DIR}

)

target\_link\_libraries(${PROJECT\_NAME}-test

${PROJECT\_NAME}

)

###################

## Code\_coverage ##

###################

find\_package(cmake\_code\_coverage QUIET)

if(cmake\_code\_coverage\_FOUND)

add\_gtest\_coverage(

TEST\_BUILD\_TARGETS

${PROJECT\_NAME}-test

)

endif()

endif()

#################

## Clang\_tools ##

#################

find\_package(cmake\_clang\_tools QUIET)

if(cmake\_clang\_tools\_FOUND)

add\_default\_clang\_tooling(

DISABLE\_CLANG\_FORMAT

)

endif(cmake\_clang\_tools\_FOUND)