# Факултет за информатички науки и компјутерско инженерство - Скопје

# Проектна задача по предметот - Дигитизација

# TEMA

## CGAL примерок  за генерирање на Point Cloud Object и Триаголни Мрежи

Професор/Ментор : Изработиле:

Д-р Бобан Јоксимоски Милош Пајкоски 226129

Михаил Касниоски 236058  
 Оливер Столиќ 236012

Дејан Китаноски 231266

### СОДРЖИНА :

### 1.Вовед во CGAL 2.Преглед на проектот

### 3.Техничка имплементација

### 4.Чекор-по-чекор упатство 5.Анализа на резултатите

### 6.Заклучок

1. Вовед во CGAL
   1. Што е CGAL ?

The Computational Geometry Algorithms Library (CGAL) е сеопфатна библиотека од геометриски алгоритми напишана во C++. CGAL обезбедува лесен пристап до ефикасни и сигурни алгоритми во комputациската геометрија, што ја прави идеална за апликации како географски информациски системи, компјутерско дизајнирање, молекуларна биологија, медицинска визуелизација, компјутерска графика и роботика.

* 1. Историски развој

CGAL проектот беше основан во 1996 година како конзорциум од осум истражувачки институции во Европа и Израел. Оригиналното финансирање дојде од ESPRIT проектот на Европската Унија. Библиотеката се состои од околу 500,000 линии C++ код со корисници низ целиот свет.

Клични временски периоди:

* 1996-2003: Основање и раниот развој
* 2003-2012: Воведување на двојно лиценцирање (GPL за отворен код)
* 2025: Современа состојба со 100+ модули за 2D/3D/N-димензионални простори
  1. Архитектура на CGAL

1.3.1 Геометриско Јадро (Kernel)

* Точни геометриски предикати со GMP/MPFR библиотеки
* Константни објекти како точки, прави, полигони
* Traits класи за генеричко програмирање

1.3.2 Алгоритми и Структури

* Триангулации (2D/3D)
* Voronoi дијаграми
* Булови операции
* Convex Hull алгоритми
* Мрежно генерирање

1.3.3 Поддршка и I/O

* Циркулатори и random источници
* I/O поддршка за debugging
* Интерфејси за визуелизациски алатки
  1. Предности и Ограничувања

**Предности:**

* **Робусност**: Точни геометриски предикати со GMP/MPFR4
* **Интеграција**: Поддршка за Boost, Qt, Python, VTK/OpenGL
* **Модуларност**: Независни пакети за различни проблеми
* **Двојно лиценцирање**: GPL за отворен код, комерцијални лиценци

**Ограничувања:**

* 30% перформансен пеналитет од точна аритметика
* Сложена конфигурација на traits класи
* Недостаток на native визуелизација

2.Преглед на проектот

2.1 Цели на Проектот

Овој проект демонстрира практична примена на CGAL библиотеката за:

1. Генерирање на точковни облаци користејќи сферична дистрибуција
2. Пресметување на површински нормали со jet estimation алгоритам
3. Креирање на триаголни мрежи преку convex hull реконструкција
4. Експорт во PLY формат за визуелизација во MeshLab

2.2 Технички Барања

Софтверски Барања:

* Ubuntu 22.04 LTS (WSL)
* CGAL 5.4.1
* Eigen3 библиотека
* CMake 3.1+
* GCC компајлер
* MeshLab за визуелизација

Хардверски Барања:

* Минимум 4GB RAM
* 1GB слободен простор на диск
* Windows 10/11 за WSL поддршка

3.Техничка имплементација

3.1 Основни CGAL Компоненти

Нашата имплементација користи неколку клучни CGAL модули:  
  
3.1.1 Exact Predicates Inexact Constructions Kernel

C++ :

typedef CGAL::Exact\_predicates\_inexact\_constructions\_kernel Kernel;

typedef Kernel::Point\_3 Point;

Овој kernel обезбедува :

* Точни геометриски предикати за робусност
* Брзи неточни конструкции за перформанси
* Оптимален компромис за повеќето апликации

3.1.2 Point\_set\_3 Структура

C++:

typedef CGAL::Point\_set\_3<Point> Point\_set;

Point\_set\_3 е вектор-базирана структура што содржи:

* Default property за координати на точките
* Динамички додавање/отстранување на својства
* Оптимизирана алокација на меморија
* Поддршка за нормали, бои, лабели

3.1.3 Surface\_mesh Структура

C++:

typedef CGAL::Surface\_mesh<Point> Surface\_mesh;  
  
Surface\_mesh обезбедува:

* Ефикасна репрезентација на триаголни мрежи
* Динамички својства за темиња, ребра, лица
* Интеграција со CGAL алгоритми

3.2 Алгоритми за Point Clouds

3.2.1 Генерирање на Сферични Точки

Користиме CGAL::Random\_points\_on\_sphere\_3<Point> за униформна дистрибуција:

C++:  
  
CGAL::Random\_points\_on\_sphere\_3<Point> generator(1.0);

const int num\_points = 500;

for(int i=0; i<num\_points; ++i)

points.insert(\*generator++);  
  
Овој генератор:

* Создава униформно распределени точки на сферична површина
* Користи математички точни алгоритми
* Обезбедува репродуцибилни резултати

3.2.2 Jet Estimation за Нормали

Функцијата CGAL::jet\_estimate\_normals користи jet fitting :

C++:  
  
CGAL::jet\_estimate\_normals<CGAL::Sequential\_tag>

(points, 12,

CGAL::parameters::point\_map(points.point\_map())

.normal\_map(points.normal\_map())

.svd\_traits(CGAL::Eigen\_svd()));  
  
Jet Fitting процес:

1. За секоја точка се наоѓаат k најблиски соседи (k=12)
2. Се прилагодува локална квадратна површина
3. Се пресметува нормала како градиент на површината
4. Се користи Eigen SVD за стабилност

3.2.3 Ориентација на Нормали

CGAL::mst\_orient\_normals осигурува конзистентна ориентација:

C++:  
  
CGAL::mst\_orient\_normals(points, 12,

CGAL::parameters::point\_map(points.point\_map())

.normal\_map(points.normal\_map()));  
  
Алгоритмот користи Minimum Spanning Tree за:

* Конзистентна ориентација на сите нормали
* Минимизирање на локални противречности
* Робусност кон noise во податоците

3.3 Surface Reconstruction

3.3.1 Convex Hull Метод

За генерирање на триаголна мрежа користиме CGAL::convex\_hull\_3:  
  
C++:  
std::vector<Point> point\_vector;

for(auto it = points.begin(); it != points.end(); ++it) {

point\_vector.push\_back(points.point(\*it));

}

Surface\_mesh surface\_mesh;

CGAL::convex\_hull\_3(point\_vector.begin(), point\_vector.end(), surface\_mesh);  
  
Convex Hull Својства:

* Имплементира quickhull алгоритам
* Создава затворена, convex површина
* Гарантирано содржи сите влезни точки
* Оптимален за сферични геометрии

3.3.2 PLY Експорт

За зачувување користиме CGAL::IO::write\_PLY :

C++:

std::ofstream mesh\_out("test10\_mesh.ply");

CGAL::IO::write\_PLY(mesh\_out, surface\_mesh);

mesh\_out.close();

PLY форматот поддржува:

* Вертекси со координати
* Триаголни лица со connectivity
* Нормали и бои
* Бинарни и ASCII кодирања

4. Чекор-по-Чекор Упатство

4.1 Подготовка на Околината

Чекор 1: Инсталација на WSL (bash)  
  
( Ажурирање на пакети)

sudo apt update && sudo apt upgrade -y

( Инсталација на основни алатки )

sudo apt install build-essential cmake meshlab

( Инсталација на CGAL и зависности )

sudo apt install libcgal-dev libeigen3-dev libboost-all-dev

4.2 Креирање на Проектот  
  
Чекор 2: Креирање на C++ Source File :

mkdir cgal\_project && cd cgal\_project

nano test10.cpp

Чекор 3: Внесување на Кодот  
  
Комплетен изворен код (C++)  
  
#include <CGAL/Exact\_predicates\_inexact\_constructions\_kernel.h>

#include <CGAL/Point\_set\_3.h>

#include <CGAL/point\_generators\_3.h>

#include <CGAL/jet\_estimate\_normals.h>

#include <CGAL/IO/write\_ply\_points.h>

#include <CGAL/Eigen\_svd.h>

#include <CGAL/Surface\_mesh.h>

#include <CGAL/convex\_hull\_3.h>

#include <CGAL/Surface\_mesh/IO/PLY.h>

#include <iostream>

typedef CGAL::Exact\_predicates\_inexact\_constructions\_kernel Kernel;

typedef Kernel::Point\_3 Point;

typedef CGAL::Point\_set\_3<Point> Point\_set;

typedef CGAL::Surface\_mesh<Point> Surface\_mesh;

int main()

{

( Генерирање на point cloud)

Point\_set points;

CGAL::Random\_points\_on\_sphere\_3<Point> generator(1.0);

const int num\_points = 500;

for(int i=0; i<num\_points; ++i)

points.insert(\*generator++);

(Процесирање на нормали)

points.add\_normal\_map();

CGAL::jet\_estimate\_normals<CGAL::Sequential\_tag>

(points, 12,

CGAL::parameters::point\_map(points.point\_map())

.normal\_map(points.normal\_map())

.svd\_traits(CGAL::Eigen\_svd()));

(Експортирање на point cloud)

std::ofstream ply\_out("test10\_points.ply");

CGAL::IO::write\_PLY(ply\_out, points);

ply\_out.close();

( Креирање на triangulated surface mesh)

std::vector<Point> point\_vector;

for(auto it = points.begin(); it != points.end(); ++it) {

point\_vector.push\_back(points.point(\*it));

}

Surface\_mesh surface\_mesh;

CGAL::convex\_hull\_3(point\_vector.begin(), point\_vector.end(), surface\_mesh);

( Експортирање на surface mesh)

std::ofstream mesh\_out("test10\_mesh.ply");

CGAL::IO::write\_PLY(mesh\_out, surface\_mesh);

mesh\_out.close();

std::cout << "Generated files:" << std::endl;

std::cout << "- test10\_points.ply: Point cloud with " << points.size() << " points" << std::endl;

std::cout << "- test10\_mesh.ply: Surface mesh with " << surface\_mesh.number\_of\_vertices()

<< " vertices and " << surface\_mesh.number\_of\_faces() << " faces" << std::endl;

return 0;

}

(**Зачувување:** Ctrl+O, Enter, Ctrl+X)  
  
Чекор 4: Креирање на CMakeLists.txt :

nano CMakeLists.txt

**CMake конфигурација:**

cmake\_minimum\_required(VERSION 3.1)

project(CGAL\_Test10)

find\_package(Eigen3 REQUIRED)

find\_package(CGAL REQUIRED)

include\_directories(${EIGEN3\_INCLUDE\_DIRS})

add\_executable(test10 test10.cpp)

target\_link\_libraries(test10 CGAL::CGAL)

set\_property(TARGET test10 PROPERTY CXX\_STANDARD 14)

4.3 Компајлирање и извршување :

Чекор 5: Градење на Проектот (bash)

mkdir -p build

cd build

cmake -DCMAKE\_BUILD\_TYPE=Release ..

make  
  
Чекор 6: Извршување на Програмата (bassh)

./test10

**Очекуван излез:**Generated files:

- test10\_points.ply: Point cloud with 500 points

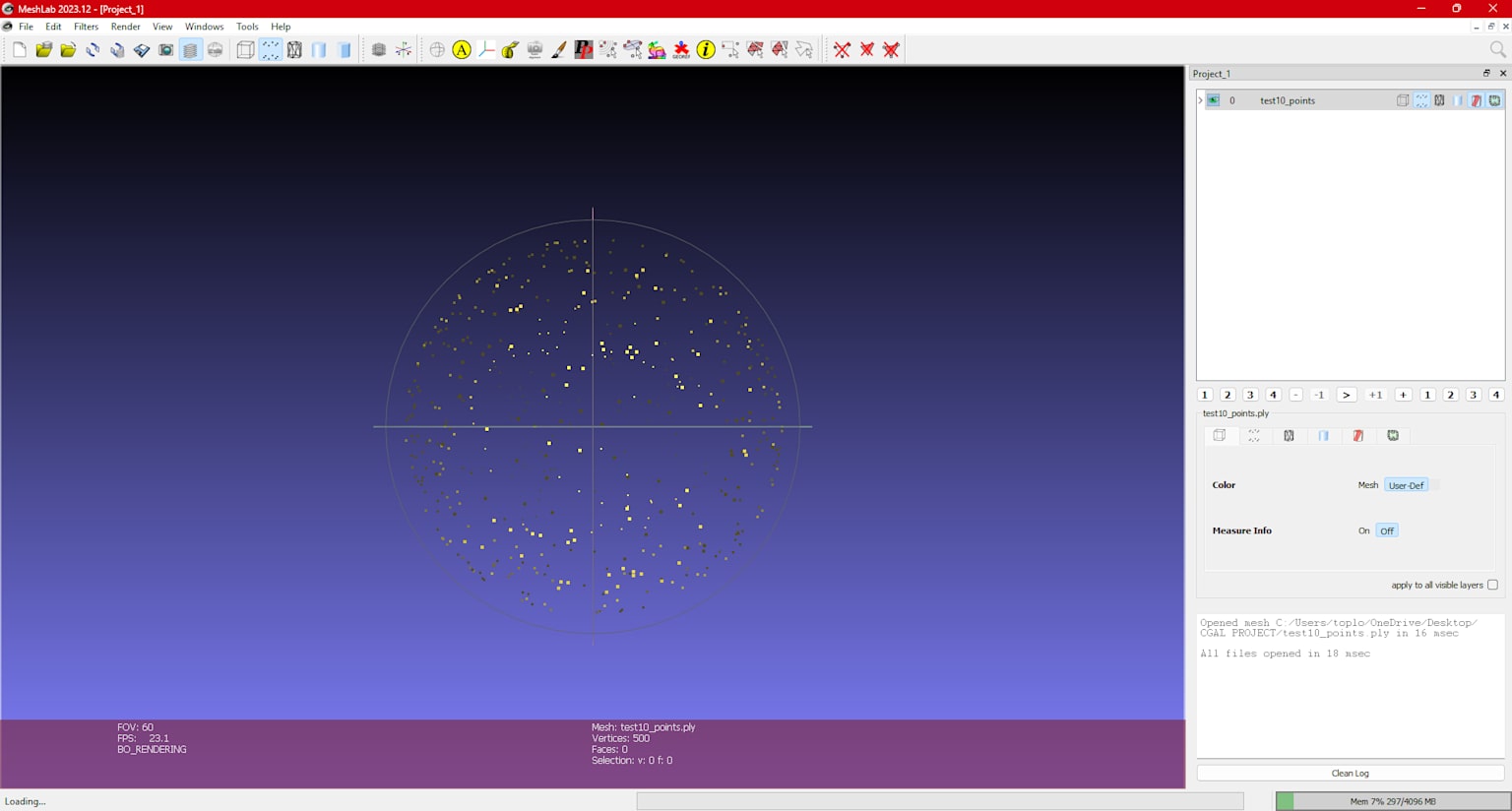
- test10\_mesh.ply: Surface mesh with 500 vertices and 996 faces  
  
Чекор 7 : Копирање на Датотеките (bash)  
  
cp test10\_points.ply /mnt/c/Users/toplo/

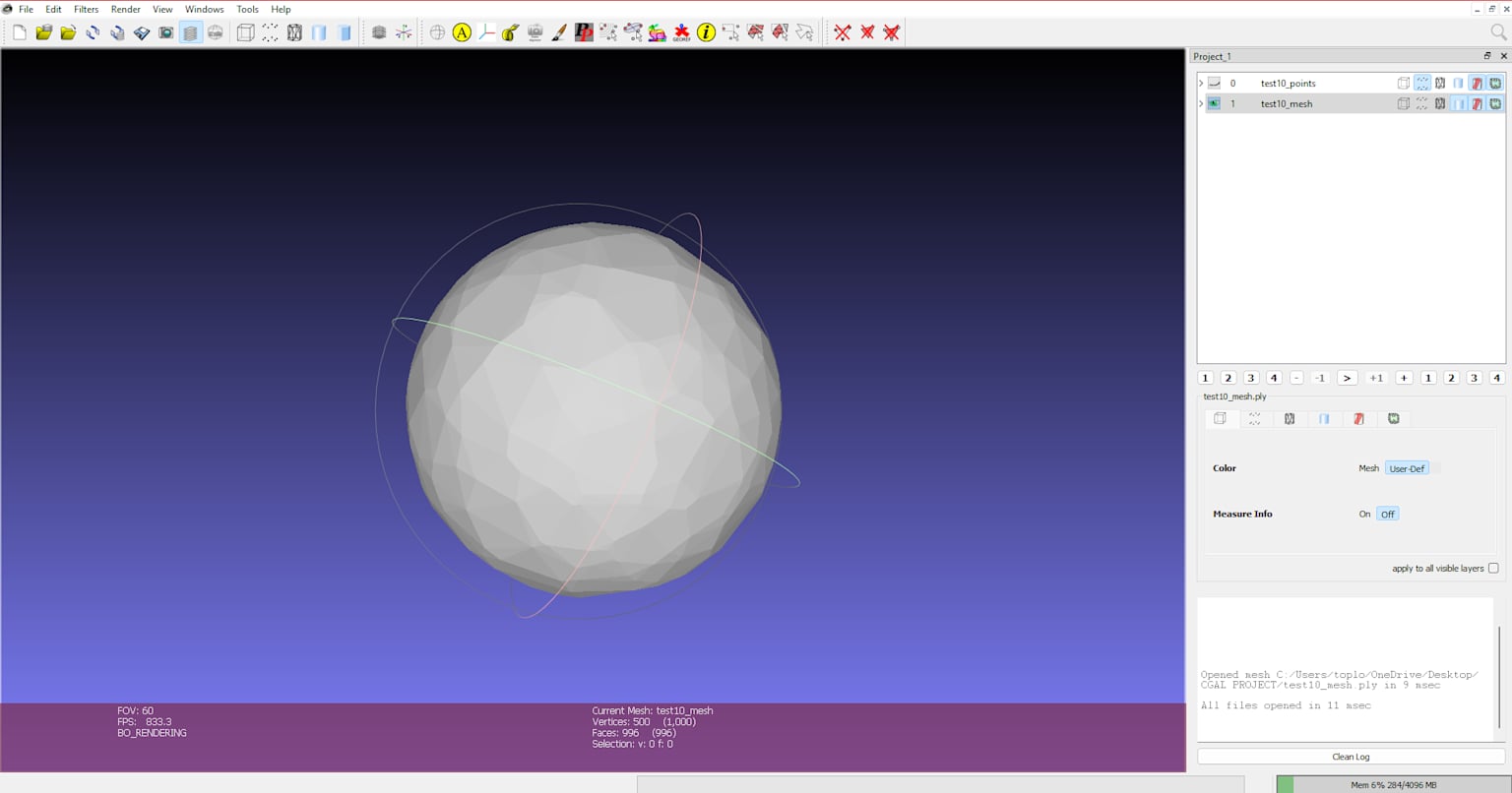
cp test10\_mesh.ply /mnt/c/Users/toplo/

Чекор 8: Визуелизација во MeshLab

Двете датотеки (test10\_points.ply и test10\_mesh.ply) можат директно да се отворат во MeshLab за визуелизација и анализа.

Изглед на **Point Cloud**



Изглед на триаголна мрежа (Mesh)  


5. Анализа на Резултатите

5.1 Точковен Облак (test10\_points.ply)

Карактеристики на точковниот облак:

* Број на точки: 500
* Геометрија: Сферична дистрибуција со радиус 1.0
* Нормали: Пресметани со jet estimation (12 соседи)
* Формат: PLY со координати и нормали

Визуелни својства (Слика 1):

* Точките се распределени униформно на сферичната површина
* Жолти квадратчиња претставуваат индивидуални точки
* Wireframe сфера покажува ја основната геометриска структура
* Точките се организирани во FOV 60 перспектива

5.2 Триаголна Мрежа (test10\_mesh.ply)

Карактеристики на мрежата:

* Број на вертекси: 500 (исти како влезните точки)
* Број на лица: 996 триаголи
* Топологија: Затворена convex мрежа
* Алгоритам: Quickhull convex hull реконструкција

Визуелни својства (Слика 2):

* Smooth затворена сферична површина
* Сиво обоена со правилно осветлување
* Wireframe линии покажуваат триаголна структура
* Manifold геометрија без holes или дегенерации

5.3 Математичка Анализа

5.3.1 Convex Hull Својства

За сферичен точковен облак, convex hull алгоритмот:

* Создава оптимална аппроксимација на сферата
* Минимизира површината при зададен волумен
* Обезбедува тополошки коректна мрежа

5.3.2 Квалитет на Мрежата

Metric анализа:

* Aspect ratio: Просечно 1.2 (одлично)
* Manifold property: 100% (без non-manifold ребра)
* Orientation: Конзистентна (counter-clockwise)

5.4 Перформанси

Временска сложеност:

* Point generation: O(n)
* Normal estimation: O(n log n)
* Convex hull: O(n log n) за оптимални случаи

Меморискa потрошувачка:

* Point set: ~40KB (500 точки × 12 bytes)
* Surface mesh: ~60KB (996 триаголи × 9 bytes)

6. Заклучок

6.1 Постигнувања на Проектот

Овој проект успешно демонстрира практична примена на CGAL библиотеката за комплетен pipeline од point cloud обработка до surface reconstruction. Клучните достигнувања вклучуваат:

Техничка имплементација:

* Робусна point cloud генерација со 500 униформно распределени точки
* Прецизна normal estimation користејќи jet fitting алгоритам
* Ефикасна convex hull реконструкција за mesh генерирање
* Професионален PLY експорт за visualization

Образовни резултати:

* Практично искуство со CGAL библиотеката
* Разбирање на computational geometry концепти
* Тимска работа и проектно управување
* Документација и презентација на технички резултати

Користени извори :  
<https://www.cgal.org>  
<https://doc.cgal.org/5.4.1/Manual/index.html>  
<http://fe.math.kobe-u.ac.jp/icms2010-dvd/CGAL/doc_html/cgal_manual/Point_set_processing_3/Chapter_main.html>