Computational Geometry

A gentle intro to CGAL library

Vissarion Fisikopoulos

National and Kapodistrian University of Athens
Dept. of Informatics and Telecommunications

2025

Geometric robustness

algorithms

- handle degeneracies
- assume real-RAM model

Geometric robustness

algorithms

- handle degeneracies
- assume real-RAM model
- computers are not real-RAMs!
- exact geometric computing paradigm

Geometric robustness

algorithms

- handle degeneracies
- assume real-RAM model
- computers are not real-RAMs!
- exact geometric computing paradigm

reference

- Kettner, Mehlhorn, Pion, Schirra, Yap. *Classroom Examples of Robustness Problems in Geometric Computations.*

Convex hull classroom example

degeneracy

- compute the convex hull of these points

$$
\begin{aligned}
p_1 &= (\ 7.3000000000000194,\ \ 7.3000000000000167\ ) \\
p_2 &= (24.000000000000068,\ 24.000000000000071\ ) \\
p_3 &= (24.00000000000005,\ \ 24.000000000000053\ ) \\
p_4 &= (\ 0.50000000000001621,\ 0.50000000000001243) \\
p_5 &= (\ 8,\ 4) \quad p_6 = (\ 4,\ 9) \quad p_7 = (15,27) \\
p_8 &= (26,25) \quad p_9 = (19,11)
\end{aligned}
$$

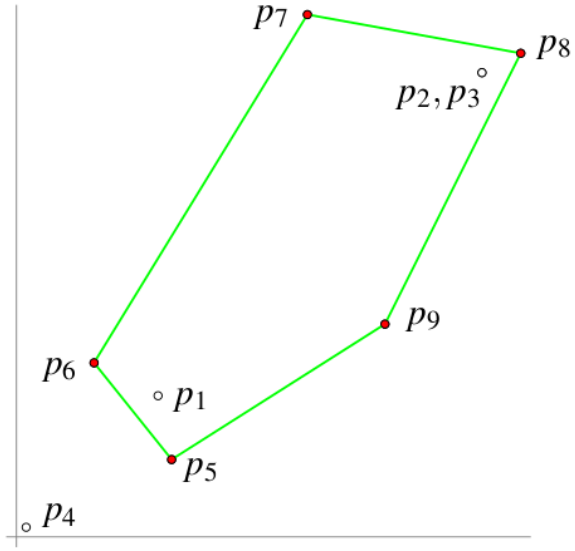
Convex hull classroom example

degeneracy

compute the convex hull of these points

$$
\begin{aligned}
p_1 &= (\ 7.3000000000000194,\ \ 7.3000000000000167\ ) \\
p_2 &= (24.000000000000068,\ 24.000000000000071\ ) \\
p_3 &= (24.00000000000005,\ \ 24.000000000000053\ ) \\
p_4 &= (\ 0.50000000000001621,\ 0.50000000000001243) \\
p_5 &= (\ 8,\ 4) \quad p_6 = (\ 4,\ 9) \quad p_7 = (15,27) \\
p_8 &= (26,25) \quad p_9 = (19,11)
\end{aligned}
$$

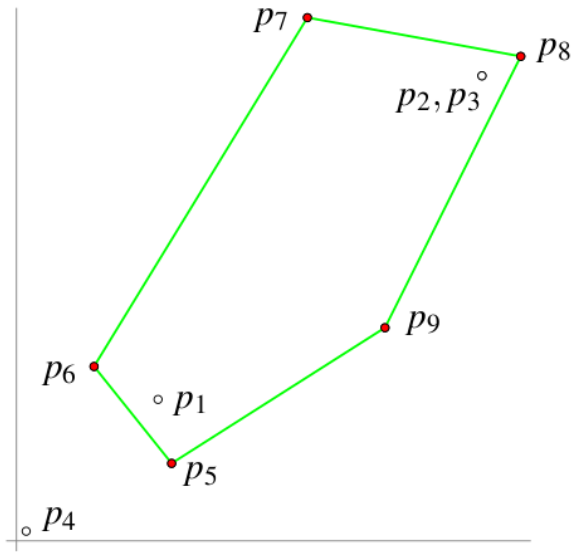
result



What happened?

inconsistency

$$float\_orient(p_1, p_2, p_3) > 0$$
$$float\_orient(p_1, p_2, p_4) > 0$$
$$float\_orient(p_2, p_3, p_4) > 0$$
$$float\_orient(p_3, p_1, p_4) > 0 \ (!!)$$



What happened?

inconsistency

$$float\_orient(p_1, p_2, p_3) > 0$$
$$float\_orient(p_1, p_2, p_4) > 0$$
$$float\_orient(p_2, p_3, p_4) > 0$$
$$float\_orient(p_3, p_1, p_4) > 0 \ (!!)$$

inacceptable

- small degeneracies
- wrong geometric result

Computational Geometry Algorithms Library

CGAL

- reference library for CG
- C++
- generic programming
- http://www.cgal.org

Computational Geometry Algorithms Library

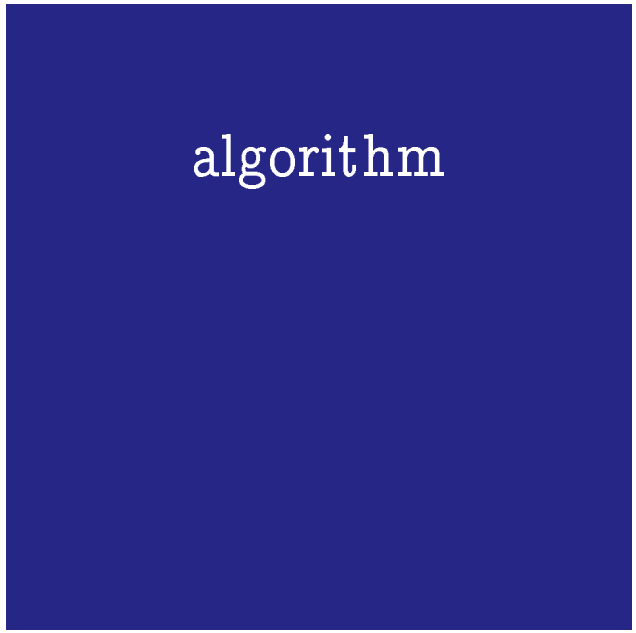
CGAL

- reference library for CG
- C++
- generic programming
- http://www.cgal.org

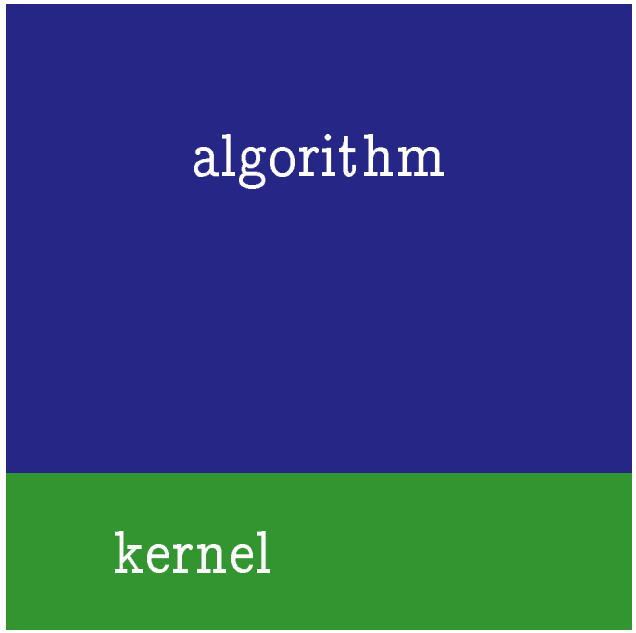
uses

- STL
- boost
- GMP and other
- Qt

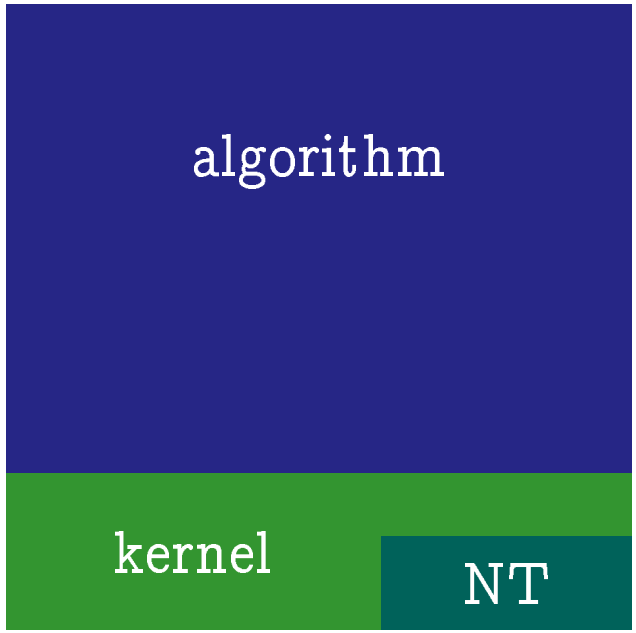
Algorithm architecture



Algorithm architecture



Algorithm architecture



Different kernels

geometric

- exact predicates, exact constructions

  `CGAL::Exact_predicates_inexact_constructions_kernel`

- exact predicates, inexact constructions

  `CGAL::Exact_predicates_exact_constructions_kernel`

- inexact

  `CGAL::Cartesian<double>`

example

```
typedef CGAL::Cartesian<double> K;
typedef K::Point_2 Point_2;
```

Different kernels

geometric

- exact predicates, exact constructions

  `CGAL::Exact_predicates_inexact_constructions_kernel`

- exact predicates, inexact constructions

  `CGAL::Exact_predicates_exact_constructions_kernel`

- inexact

  `CGAL::Cartesian<double>`

algebraic

- univariate
- bivariate

How to reach robustness?

use CGAL

- exact kernels
- choose the best kernel
- forget about the details

How to reach robustness?

use CGAL

- exact kernels
- choose the best kernel
- forget about the details

reuse

- same code
- swap kernels
- change number types
- different numeric results
- same geometric results

The standard template library (STL)

containers

- instead of arrays
- vector, list, ...

The standard template library (STL)

containers

- instead of arrays
- vector, list, ...

example

```
vector<int> v(3);   // Declare a vector of 3 elements.
v[0] = 7;
v[1] = v[0] + 3;
v[2] = v[0] + v[1];
```

$v[0] == 7, v[1] == 10, v[2] == 17$

The standard template library (STL)

iterators

- instead of indices
- forward, backward, insert iterators

The standard template library (STL)

iterators

- instead of indices
- forward, backward, insert iterators

example

```
vector<int> v(3);  // Declare a vector of 3 elements.
v[0] = 7;
v[1] = v[0] + 3;
v[2] = v[0] + v[1];

typedef std::vector<int>::iterator Viterator;
for (Viterator it=v.begin(); it!=v.end(); ++it)
  std::cout << *it << std::endl;
```

$v[0] == 7, v[1] == 10, v[2] == 17$

The standard template library (STL)

advantages

- reuse code
- enhance code readability

more info

http://www.sgi.com/tech/stl

STL example

example

```
#include <iostream>
#include <vector>

typedef std::vector<double>              dvector;
typedef std::vector<double>::iterator    dveciterator;

int main()
{
  // construct a vector
  dvector p;

  // insert elements into vector
  p.push_back(0);
  p.push_back(10);
  p.push_back(10);
  p.push_back(6);
  p.push_back(4);

  // iterate through vector's elements and print them
  for (dveciterator it=p.begin(); it!= p.end(); ++it)
    std::cout << *it << std::endl;

  return 0;
}
```

CGAL examples

```
#include <CGAL/
      Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/convex_hull_2.h>

#include <vector>

typedef CGAL::
      Exact_predicates_inexact_constructions_kernel K;
typedef K::Point_2 Point_2;
typedef std::vector<Point_2> Points;

int main()
{
   Points points, result;
   points.push_back(Point_2(0,0));
   points.push_back(Point_2(10,0));
   points.push_back(Point_2(10,10));
   points.push_back(Point_2(6,5));
   points.push_back(Point_2(4,1));

   // compute convex hull
   CGAL::convex_hull_2(points.begin(), points.end(), std::
         back_inserter(result));

   // print points of the convex hull
   for (std::vector<Point_2>::iterator it=result.begin();
         it!= result.end(); ++it)
     std::cout << *it << std::endl;

   return 0;
}
```