Cours d'Algorithmique Avancée (INFO036), Université de Liège

6th February 2007

Introduction to Convex Hull Applications

Cyril Briquet

Department of EE & CS
University of Liège, Belgium

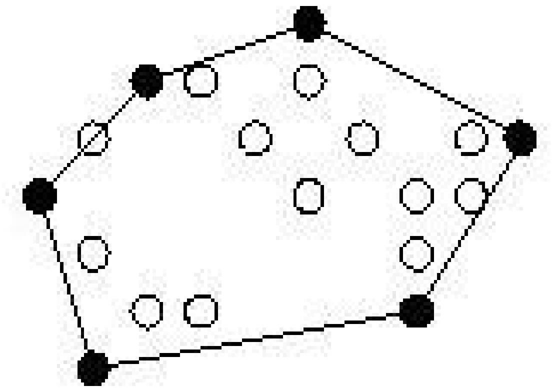
Overview

- **Convex Hull – basic notions**
- Convex Hull – application domains
- Onion Peeling – basic notions
- Onion Peeling – application domains
- Overview of classic algorithms
- Integration of a Convex Hull algorithm

Convex Hull - basic notions (I)

- input: set of *N* sites
  (i.e. data points in 2, 3,... dimensions)

- Convex Hull (2D):
  smallest enveloping polygon
  of the *N* sites



- output: ordered subset of *h* sites

Convex Hull - basic notions (II)

- relationship with sorting

- worst-case computational complexity:
  output-independent - $O(N^2)$, $O(N \log N)$
  output-sensitive - $O(N \log h)$

- storage requirements: $O(N)$, in situ

Overview

- Convex Hull – basic notions
- **Convex Hull – application domains**
- Onion Peeling – basic notions
- Onion Peeling – application domains
- Overview of classic algorithms
- Integration of a Convex Hull algorithm

Convex Hull – application domains

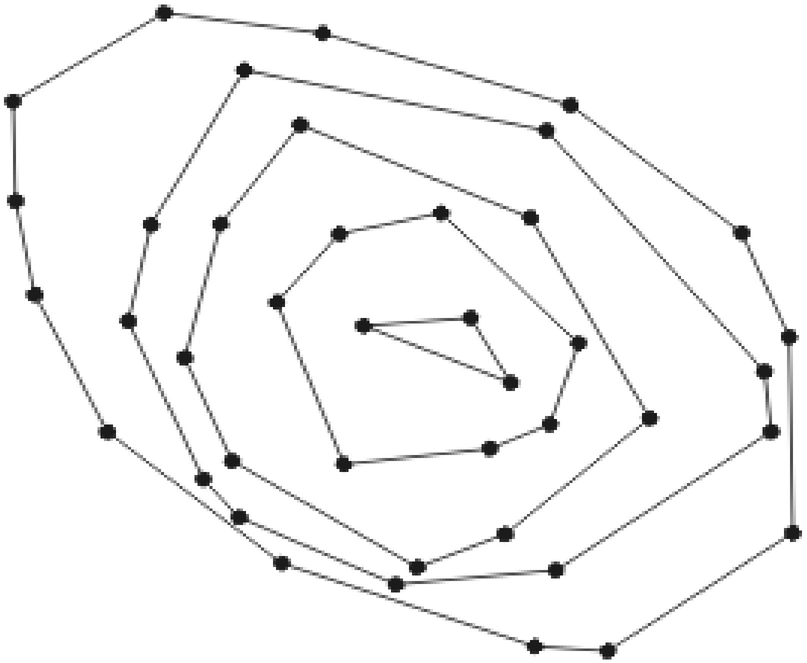
- computer visualization, ray tracing
  (e.g. video games, replacement of bounding boxes)
- path finding
  (e.g. embedded AI of Mars mission rovers)
- Geographical Information Systems (GIS)
  (e.g. computing accessibility maps)
- visual pattern matching
  (e.g. detecting car license plates)
- verification methods
  (e.g. bounding of Number Decision Diagrams)
- geometry
  (e.g. diameter computation)

Overview

- Convex Hull – basic notions
- Convex Hull – application domains
- **Onion Peeling – basic notions**
- Onion Peeling – application domains
- Overview of classic algorithms
- Integration of a Convex Hull algorithm

Onion Peeling - basic notions (I)

- Onion Peeling: sequence of nested convex hulls



- computational complexity: also O(*N* log *N*)

Overview

- Convex Hull – basic notions
- Convex Hull – application domains
- Onion Peeling – basic notions
- **Onion Peeling – application domains**
- Overview of classic algorithms
- Integration of a Convex Hull algorithm

Onion Peeling – application domains (I)

- propagation of chemical events:
  preprocessing to enable *depth retrieval*
- robust statistical estimators:
  detection of *outliers*
- study of Earth atmosphere
- network protocols (CDMA)

Onion Peeling – application domains (II)

HALogen Occultation Experiment (HALOE, NASA)

- Earth atmosphere profiling via solar occultation
- *« limb viewing experiment: measurements of the atmosphere from the UARS satellite, along paths tangents to Earth surface »*

(limb = outermost edge of a celestial body)

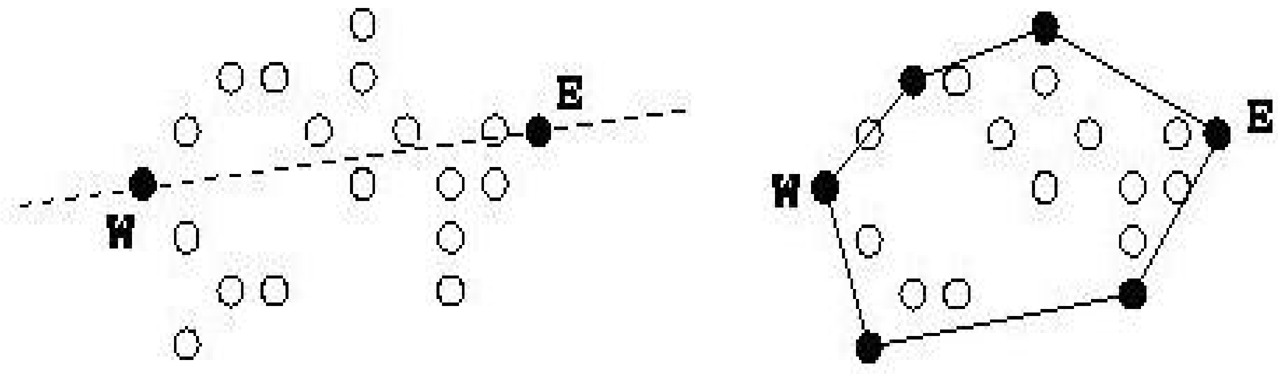
- layers of the atmosphere = Convex Hulls

Overview

- Convex Hull – basic notions
- Convex Hull – application domains
- Onion Peeling – basic notions
- Onion Peeling – application domains
- **Overview of classic algorithms**
- Integration of a Convex Hull algorithm

Overview of classic algorithms

- some Convex Hull algorithms require that input data is preprocessed:
  sites are sorted by lexicographical order
  (by X coordinate, then Y coordinate for equal X)
- most Convex Hull algorithms are designed to operate on a half plane



- E, W: extremal sites in lexicographical order

Overview of classic algorithms

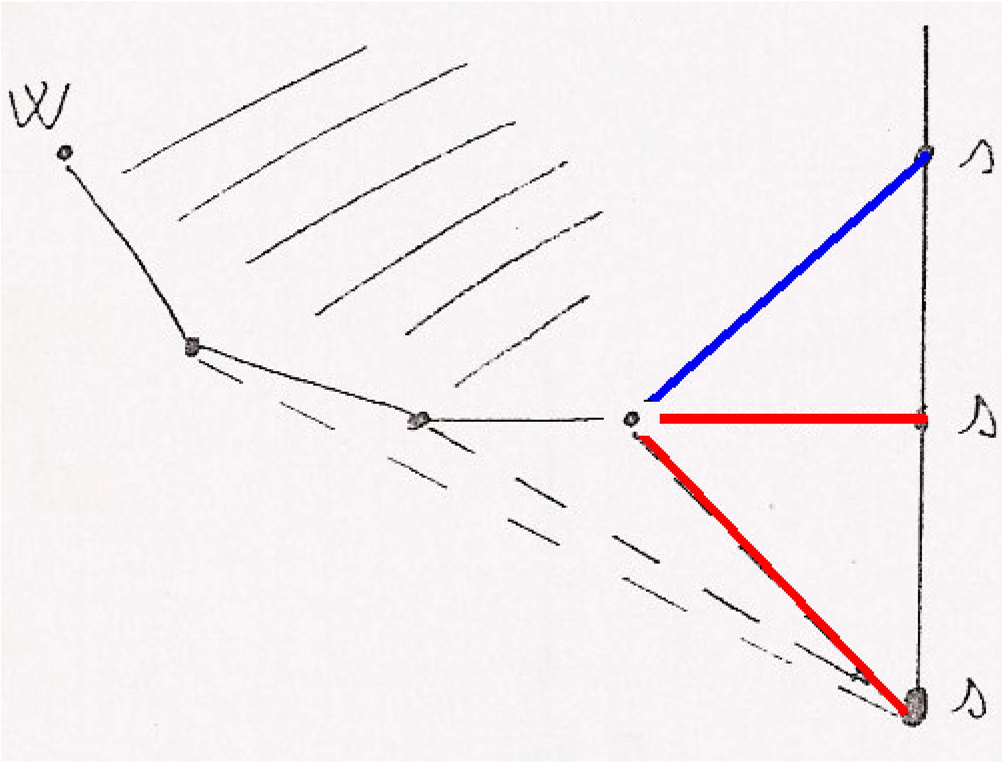
- Sort Hull (*Marche de Graham*) – requires preprocessing
- WrapHull (*Marche de Jarvis*)
- BridgeHull – requires preprocessing
- MergeHull – uses SortHull
- QuickHull

Overview of classic algorithms

Sort Hull

- process sites in lexicographical order
- for each site s, determine if last site of partial Convex Hull should be kept or removed

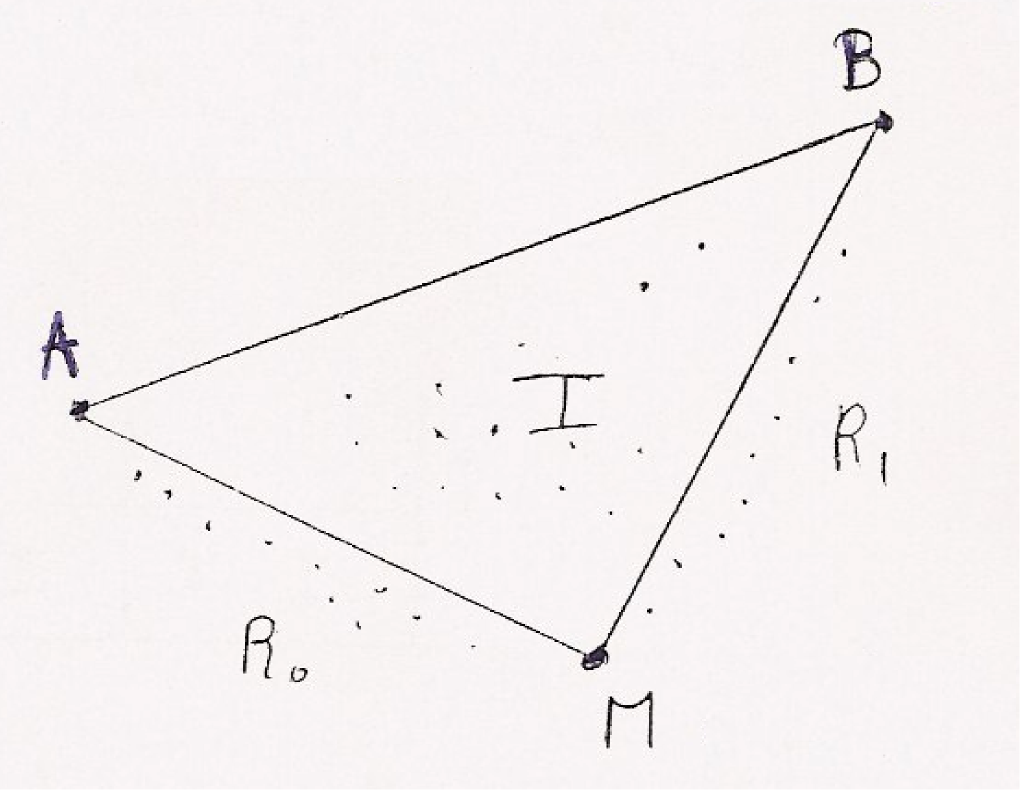
  (if removed, reevaluate last site of partial CH)



Overview of classic algorithms

QuickHull

- select pivot M, partition space into 3 sets R0, R1, I
- A, B, M included in the Convex Hull
- apply again to R0, R1

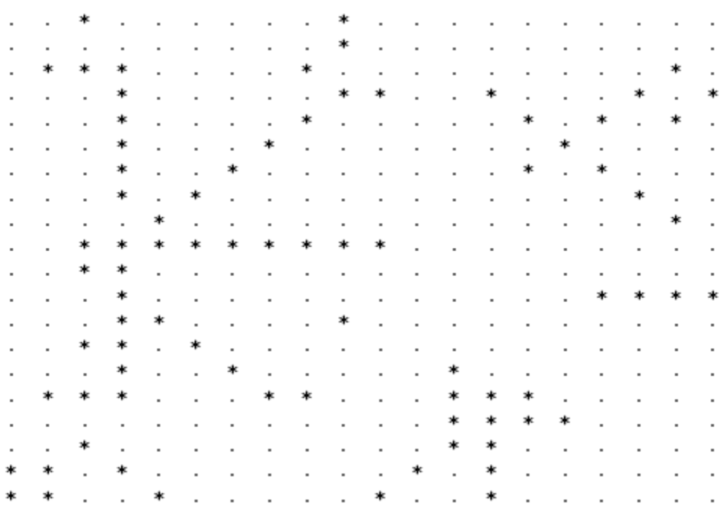


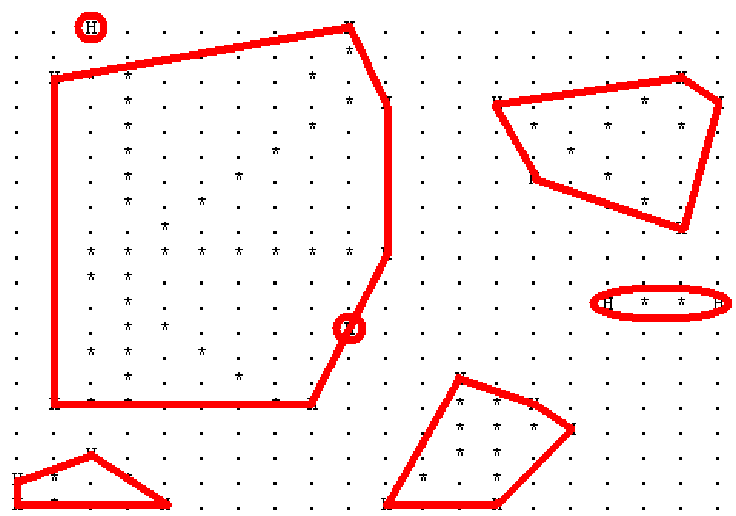
Overview

- Convex Hull – basic notions
- Convex Hull – application domains
- Onion Peeling – basic notions
- Onion Peeling – application domains
- Overview of classic algorithms
- **Integration of a Convex Hull algorithm**

Integration of a Convex Hull algorithm

- GIS problem: from satellite imagery, compute the convex hulls of a set of *barriers*
- Input: a matrix of booleans





- Output: 24 sites ordered in 7 Convex Hulls

Integration of a Convex Hull algorithm

- QuickHull is the fastest Convex Hull algorithm
- ... or is it ?

- it was noticed that in this setup,
  SortHull is known to perform better
  - small number of sites in each Convex Hull
  - most Convex Hulls are long and thin

    (e.g. roads, rivers, human-made barriers)

Integration of a Convex Hull algorithm

- but SortHull requires preprocessing anyways, so all gain over QuickHull would be lost ...

=> considering the Convex Hull algorithm
within the context of the chain of algorithms
needed to solve this problem
led to an efficient solution

Integration of a Convex Hull algorithm

0, 1, 2, 3, 4, 5, 6 = regions of connected sites

```
.  .  0  .  .  .  .  .  .  1  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  1  .  .  .  .  .  .  .  .  .  .
.  1  1  1  .  .  .  .  1  .  .  .  .  .  .  .  .  .  2  .
.  .  .  1  .  .  .  .  .  1  1  .  .  2  .  .  .  2  .  2
.  .  .  1  .  .  .  .  1  .  .  .  .  .  2  .  2  .  2  .
.  .  .  1  .  .  .  1  .  .  .  .  .  .  .  2  .  .  .  .
.  .  .  1  .  .  1  .  .  .  .  .  .  .  2  .  2  .  .  .
.  .  .  1  .  1  .  .  .  .  .  .  .  .  .  .  .  2  .  .
.  .  .  .  1  .  .  .  .  .  .  .  .  .  .  .  .  .  2  .
.  .  1  1  1  1  1  1  1  1  1  .  .  .  .  .  .  .  .  .
.  .  1  1  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  1  .  .  .  .  .  .  .  .  .  .  .  .  3  3  3  3
.  .  .  1  1  .  .  .  .  4  .  .  .  .  .  .  .  .  .  .
.  .  1  1  .  1  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  1  .  .  1  .  .  .  .  .  5  .  .  .  .  .  .  .
.  1  1  1  .  .  .  1  1  .  .  .  5  5  5  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  5  5  5  5  .  .  .  .
.  .  6  .  .  .  .  .  .  .  .  .  5  5  .  .  .  .  .  .
6  6  .  6  .  .  .  .  .  .  .  5  .  5  .  .  .  .  .  .
6  6  .  .  6  .  .  .  .  .  5  .  .  5  .  .  .  .  .  .
```

Integration of a Convex Hull algorithm

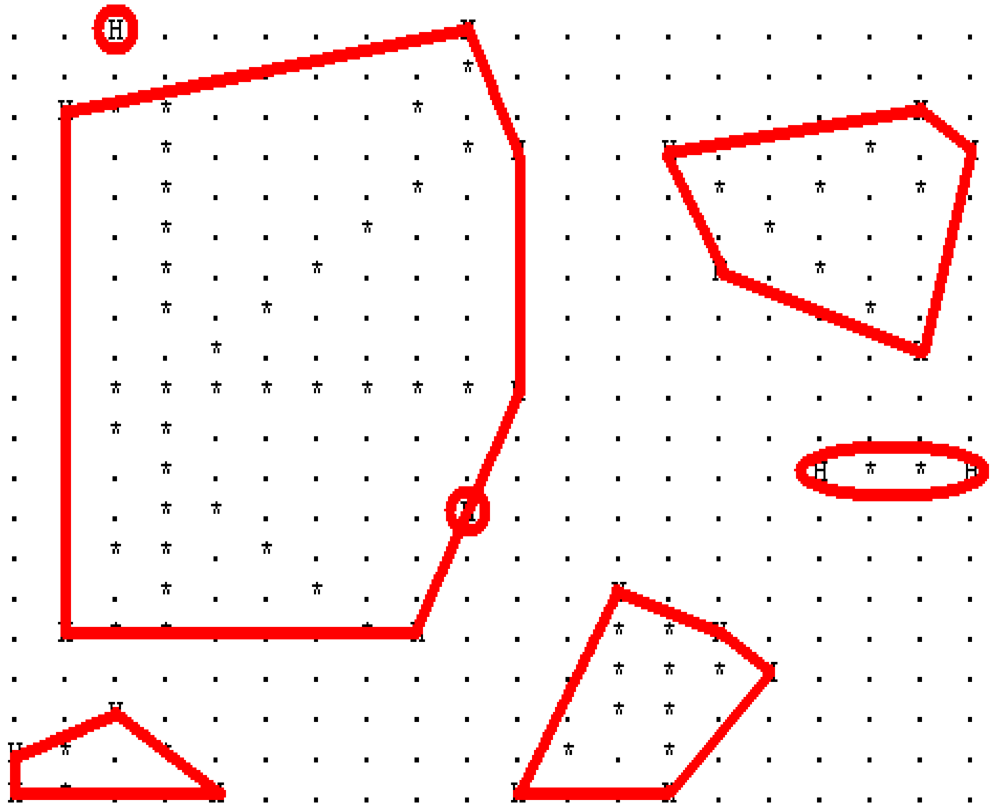
L = Lower, U = Upper, bound of a discrete *bar*

```
.  .  B  .  .  .  .  .  .  U  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  *  .  .  .  .  .  .  .  .  .  .
.  U  U  U  .  .  .  .  U  .  .  .  .  .  .  .  .  .  U  .
.  .  .  *  .  .  .  .  .  *  U  .  .  B  .  .  .  U  .  B
.  .  .  *  .  .  .  .  *  .  .  .  .  .  U  .  U  .  *  .
.  .  .  *  .  .  .  U  .  .  .  .  .  .  .  B  .  .  .  .
.  .  .  *  .  .  U  .  .  .  .  .  .  .  L  .  L  .  .  .
.  .  .  *  .  U  .  .  .  .  .  .  .  .  .  .  .  L  .  .
.  .  .  .  U  .  .  .  .  .  .  .  .  .  .  .  .  .  L  .
.  .  *  *  *  *  *  *  *  L  L  .  .  .  .  .  .  .  .  .
.  .  *  *  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  *  .  .  .  .  .  .  .  .  .  .  .  .  B  B  B  B
.  .  .  *  L  .  .  .  .  B  .  .  .  .  .  .  .  .  .  .
.  .  *  *  .  L  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.  .  .  *  .  .  L  .  .  .  .  .  U  .  .  .  .  .  .  .
.  L  L  L  .  .  .  L  L  .  .  .  *  U  U  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .  *  *  L  B  .  .  .  .
.  .  B  .  .  .  .  .  .  .  .  .  L  *  .  .  .  .  .  .
U  U  .  B  .  .  .  .  .  .  .  B  .  *  .  .  .  .  .  .
L  L  .  .  B  .  .  .  .  .  B  .  .  L  .  .  .  .  .  .
```

at this point, * sites can be filtered out

Integration of a Convex Hull algorithm

Sort Hull is applied and keeps 24 sites



Integration of a Convex Hull algorithm

- computational complexity is very important
- selecting an algorithm only on its complexity may lead to suboptimal performance of the whole chain of algorithms it belongs to