Describing and implementing basic geometric objects

Vera Sacristán

Discrete and Algorithmic Geometry
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

Basic geometric objects

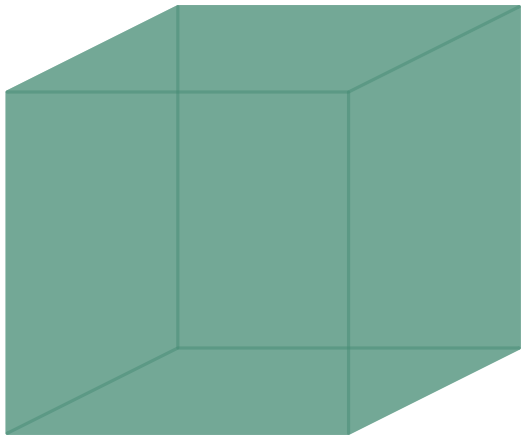
- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$

Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

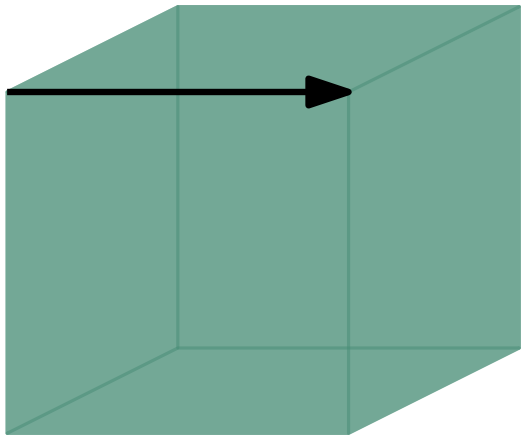
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

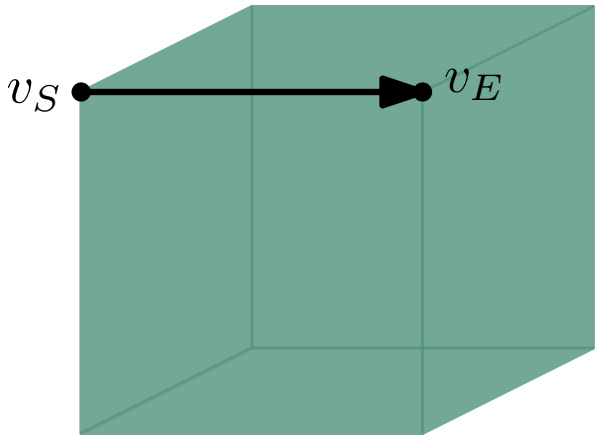
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

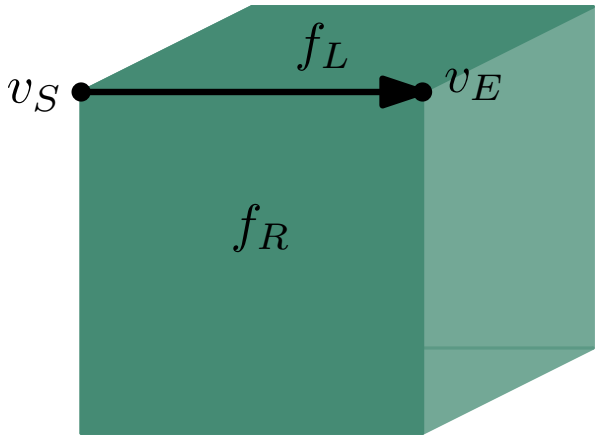
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

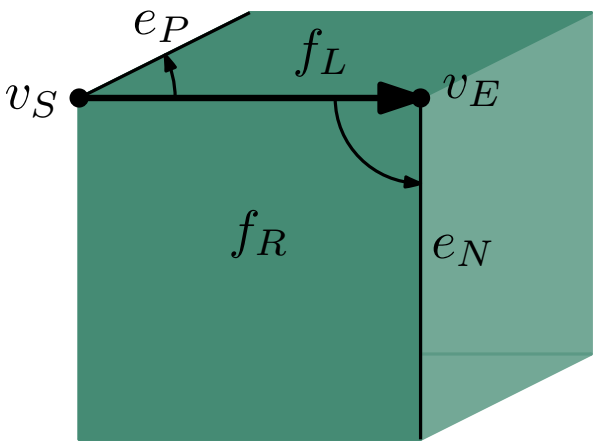
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



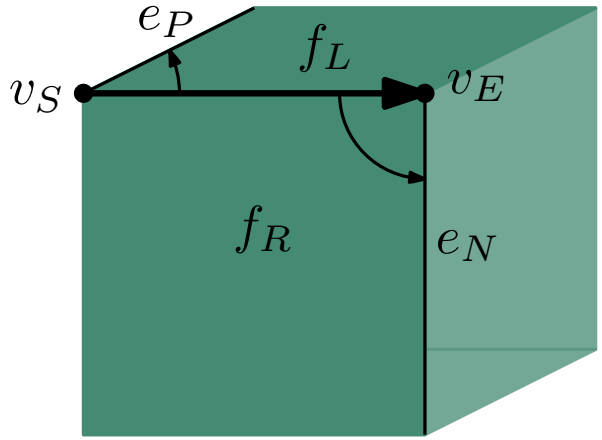
Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$

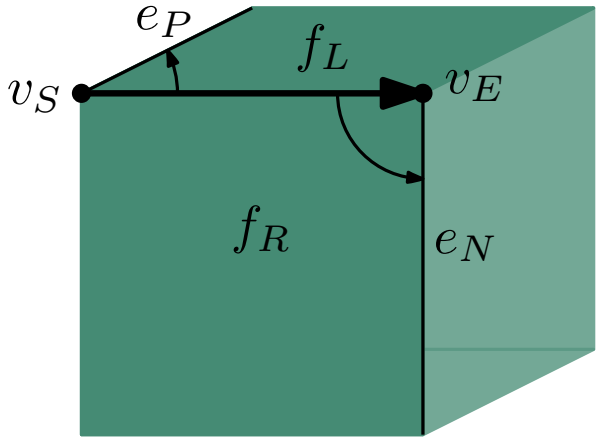


Why do we use a DCEL?



Why do we use a DCEL?

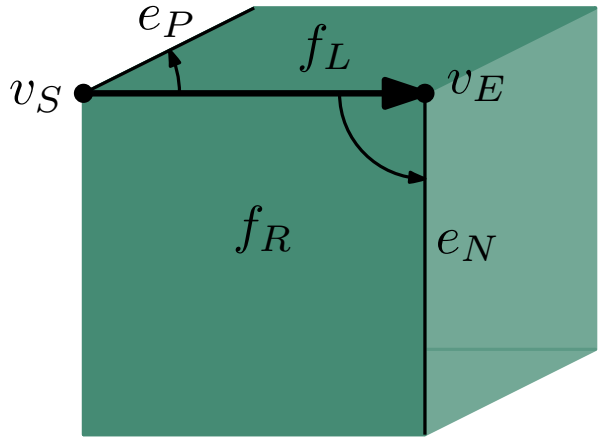
- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.



- Space:
  - The space it occupies is linear in the number of vertices:

Why do we use a DCEL?

- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.



- Space:
  - The space it occupies is linear in the number of vertices:

Proof: Since the graph is planar, $V + F = E + 2$.
Since the faces are n-gons (with $n \geq 3$), $2E \geq 3F$.

Therefore:

$$2V + 2F = 2E + 4 \geq 3F + 4 \Longrightarrow F \leq 2V - 4.$$

$$3E + 6 = 3V + 3F \leq 3V + 2E \Longrightarrow E \leq 3V - 6.$$

How to deal with real numbers in a robust way?

How to deal with real numbers in a robust way?

$\mathbb{N}$ and $\mathbb{Z}$ are easy to deal with.

$\mathbb{Q}$ is $\mathbb{Z} \times \mathbb{Z}/\sim$, where $(a, b) \sim (a', b') \iff ab' = a'b$.

Algebraic numbers in $\mathbb{R}$ can be symbollically treated, but trascendental cannot.

$\mathbb{C}$ can be identified with $\mathbb{R}^2$.

$\mathbb{H}$ can be identified with $\mathbb{R}^4$.

How to deal with real numbers in a robust way?

$\mathbb{N}$ and $\mathbb{Z}$ are easy to deal with.

$\mathbb{Q}$ is $\mathbb{Z} \times \mathbb{Z}/\sim$, where $(a, b) \sim (a', b') \iff ab' = a'b$.

Algebraic numbers in $\mathbb{R}$ can be symbollically treated, but trascendental cannot.

$\mathbb{C}$ can be identified with $\mathbb{R}^2$.

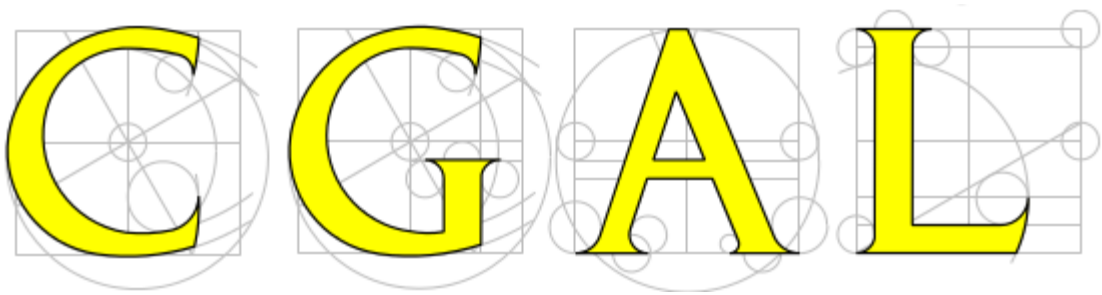
$\mathbb{H}$ can be identified with $\mathbb{R}^4$.

Recommended libraries

The GNU multiple-precision arithmetic library: https://gmplib.org/

The GNU multiple-precision floating-point library: http://www.mpfr.org/

The Computational Geometry Algorithms Library: http://www.cgal.org/



Homogeneous coordinates are very useful in practice!

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0$$
$$\iff p \perp \ell$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0$$
$$\iff p \perp \ell$$

Line through two points

$$\left.\begin{array}{l} p \text{ lies in } \ell \Longrightarrow p \perp \ell \\ q \text{ lies in } \ell \Longrightarrow q \perp \ell \end{array}\right\} \Longrightarrow \ell = p \times q$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0$$
$$\iff p \perp \ell$$

Line through two points

$$\left.\begin{array}{l} p \text{ lies in } \ell \Longrightarrow p \perp \ell \\ q \text{ lies in } \ell \Longrightarrow q \perp \ell \end{array}\right\} \Longrightarrow \ell = p \times q$$

Intersecting two lines

$$\left.\begin{array}{l} p \text{ lies in } \ell_1 \Longrightarrow p \perp \ell_1 \\ p \text{ lies in } \ell_2 \Longrightarrow p \perp \ell_2 \end{array}\right\} \Longrightarrow p = \ell_1 \times \ell_2$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0$$
$$\iff p \perp \ell$$

Line through two points

$$\left.\begin{array}{l} p \text{ lies in } \ell \Longrightarrow p \perp \ell \\ q \text{ lies in } \ell \Longrightarrow q \perp \ell \end{array}\right\} \Longrightarrow \ell = p \times q$$

Intersecting two lines

$$\left.\begin{array}{l} p \text{ lies in } \ell_1 \Longrightarrow p \perp \ell_1 \\ p \text{ lies in } \ell_2 \Longrightarrow p \perp \ell_2 \end{array}\right\} \Longrightarrow p = \ell_1 \times \ell_2$$

Parallel lines

$$\ell_1 \| \ell_2 \Longrightarrow \left\{\begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, d) \end{array}\right\} \Longrightarrow \ell_1 \times \ell_2 = (x, y, 0)$$

Homogeneous coordinates are very useful in practice!

If you implement the cross product, you get a code that is:

- **Correct:** it computes exactly what it should.
- **Reusable:** it is used to solve several (apparently different) problems.
- **Efficient:** 6 products and 3 additions.
- **Robust:** it handles all cases, including degeneracies.

Homogeneous coordinates are very useful in practice!

If you implement the cross product, you get a code that is:

- **Correct:** it computes exactly what it should.
- **Reusable:** it is used to solve several (apparently different) problems.
- **Efficient:** 6 products and 3 additions.
- **Robust:** it handles all cases, including degeneracies.

There is no way you can do the same using affine coordinates!

FURTHER READING

See the documents provided at the end of this course