Dealing with numbers, points, and lines

Vera Sacristán

Geometric Tools for Computer Graphics
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

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 $\mathbb{R}^4$.

DEALING WITH NUMBERS, POINTS, AND LINES

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 $\mathbb{R}^4$.

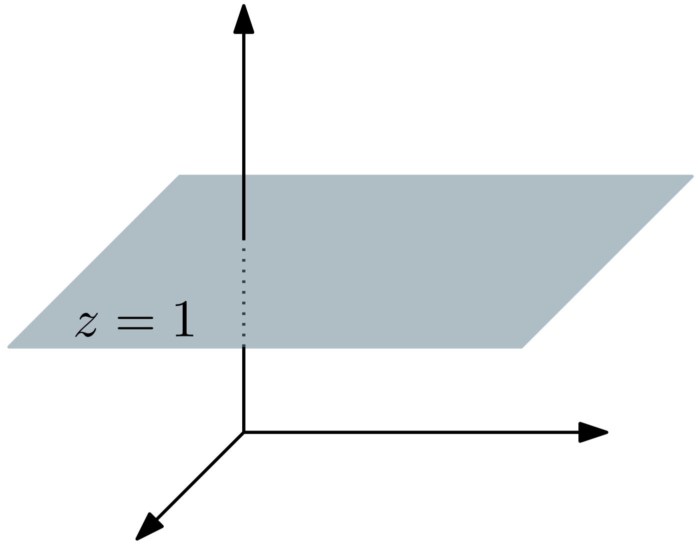
Recommended library

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

Homogeneous coordinates

Homogeneous coordinates

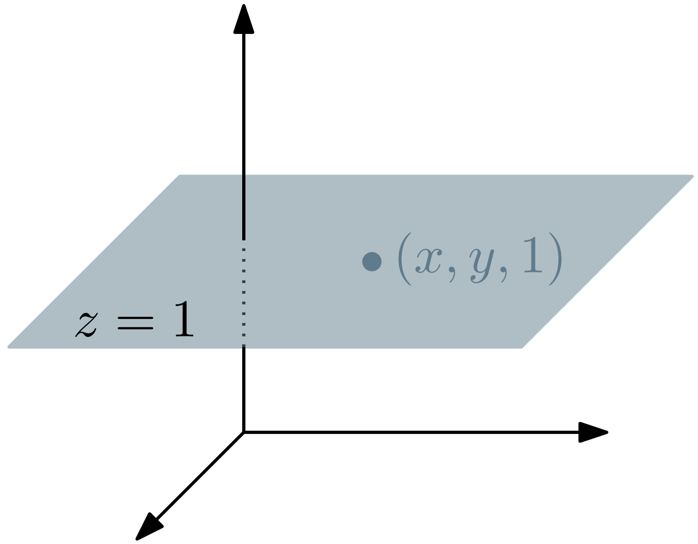
Embed the plane in 3-dimensional space as plane $z = 1$.



Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.



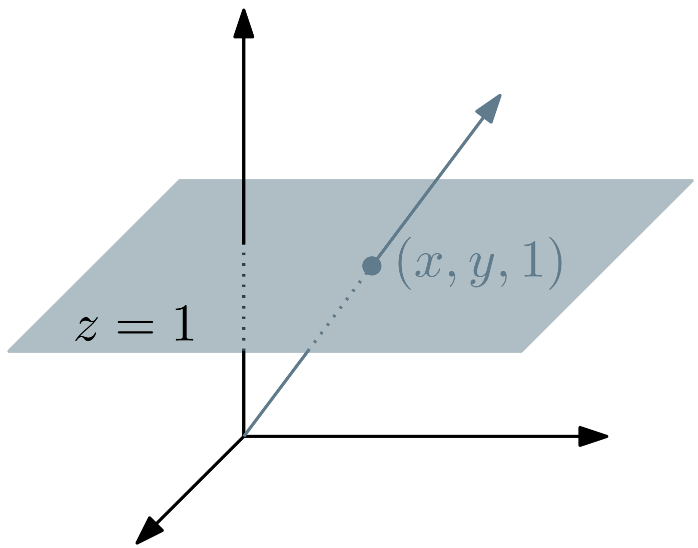
Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.



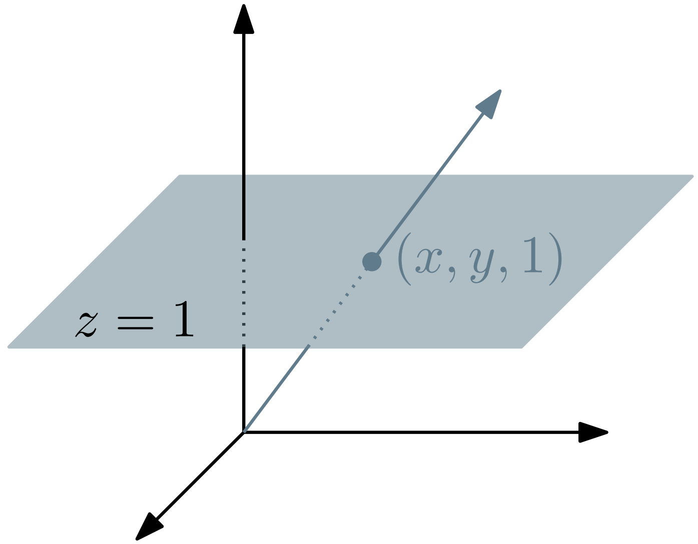
Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.



Welcome to **projective geometry**!

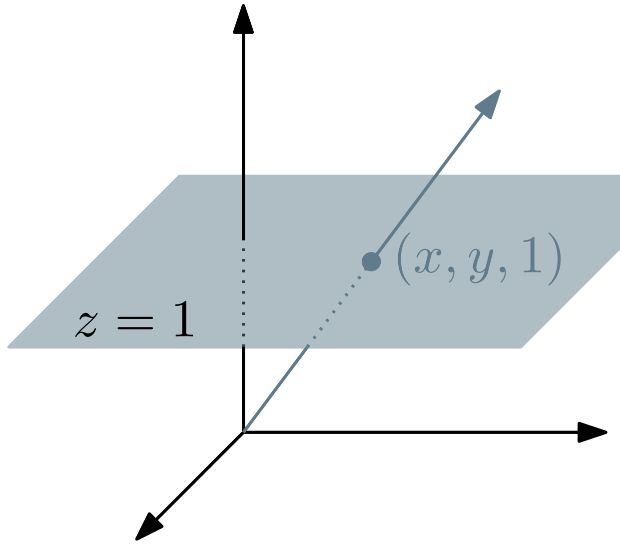
Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.



Welcome to **projective geometry**!

What about vectors?

- Vectors have coordinates $(x : y : 0)$
  because they are parallel to the plane $z = 1$.
- They correspond to horizontal rays, i.e., to points at infinity.

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

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

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

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

Intersecting two lines

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

Homogeneous coordinates are very useful in practice!

Parallel lines

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

Homogeneous coordinates are very useful in practice!

Parallel lines

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

The line at infinity

$$(x_1 : y_1 : 0) \times (x_2 : y_2 : 0) = (0 : 0 : 1)$$

Homogeneous coordinates are very useful in practice!

Parallel lines

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

The line at infinity

$$(x_1 : y_1 : 0) \times (x_2 : y_2 : 0) = (0 : 0 : 1)$$

Identical lines (or points)

$$\ell_1 = \ell_2 \Longleftrightarrow \left\{ \begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, \lambda c) \end{array} \right\} \Longleftrightarrow \ell_1 \times \ell_2 = (0, 0, 0)$$

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!