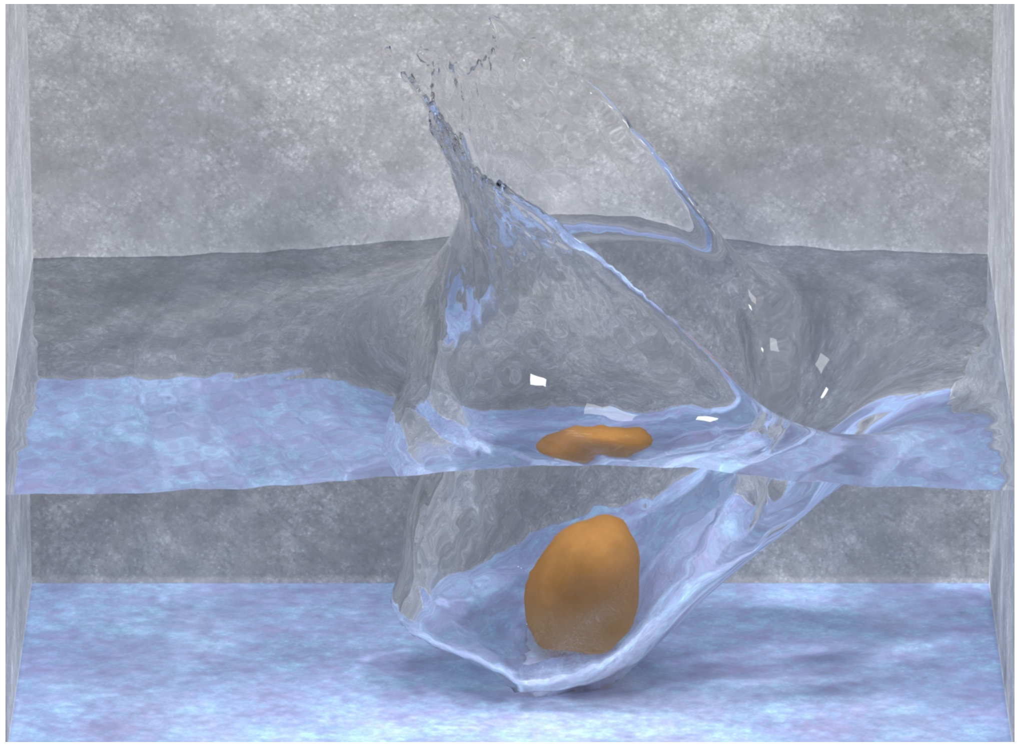
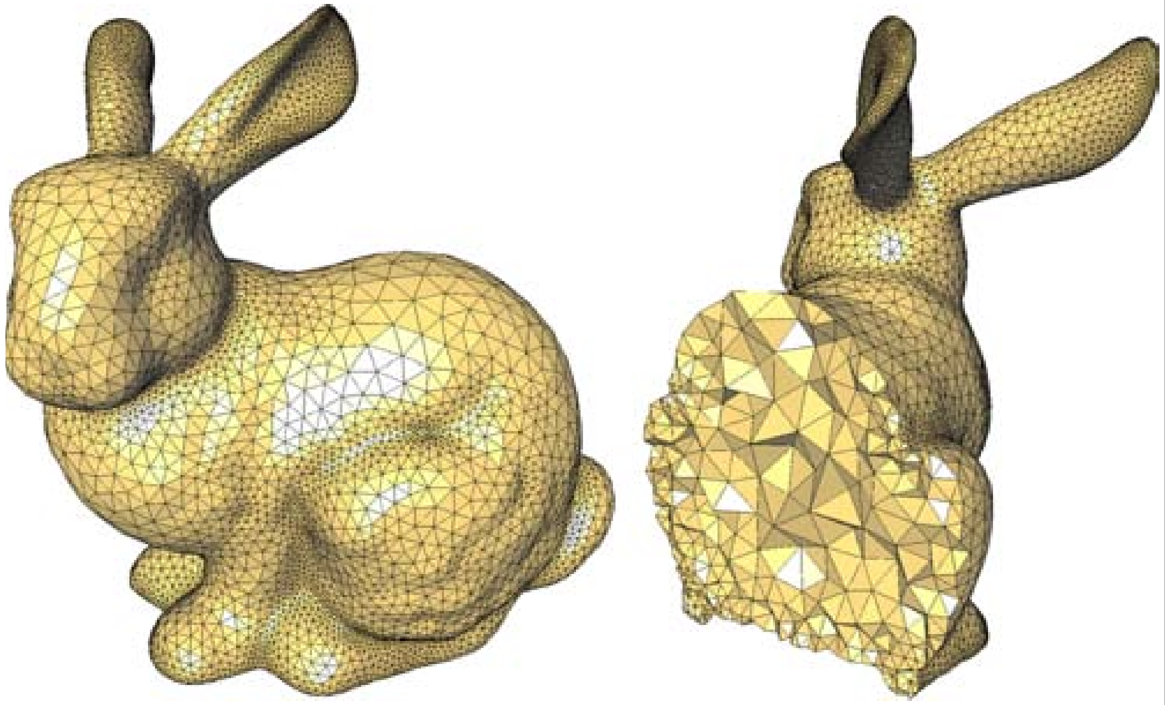
Discrete representations of geometric objects: Features, data structures and adequacy for dynamic simulation:

Part II : Levelsets & implicit surfaces





Review

Review

- *Meshed objects* are composed of 2 parts:
    - An array of *particles* (with "attributes" such as position, velocity, mass, etc)
    - A *mesh* data structure, encoded as an array of segments, triangles, tetrahedra, etc
      (whose vertices are the predefined particles)

Review

- *Meshed objects* are composed of 2 parts:
  - An array of *particles* (with "attributes" such as position, velocity, mass, etc)
  - A *mesh* data structure, encoded as an array of segments, triangles, tetrahedra, etc
    (whose vertices are the predefined particles)
- Topological queries & Derivative structures
  - ✓ Can be precomputed, do not need to store explicitly

Review

- *Meshed objects* are composed of 2 parts:
  - An array of *particles* (with "attributes" such as position, velocity, mass, etc)
  - A *mesh* data structure, encoded as an array of segments, triangles, tetrahedra, etc
    (whose vertices are the predefined particles)
- Topological queries & Derivative structures
  - ✓ Can be precomputed, do not need to store explicitly
- Geometrical queries (collisions, inside/outside tests)
  - ✓ Cannot be precomputed, since they depend on the particle attribute values
  - ✓ Potentially expensive to determine

Implicit curves and surfaces (aka. Level-Sets)

- Motivation
  - ✓ Accelerated geometric queries for problems such as:
    - ➡ Is a point (x*,y*) *inside* the object?
    - ➡ Is a point (x*,y*) *within a distance of d** from the object surface?
    - ➡ What is the point on the surface which is *closest* to the query point (x*,y*)?

Implicit curves and surfaces (aka. Level-Sets)

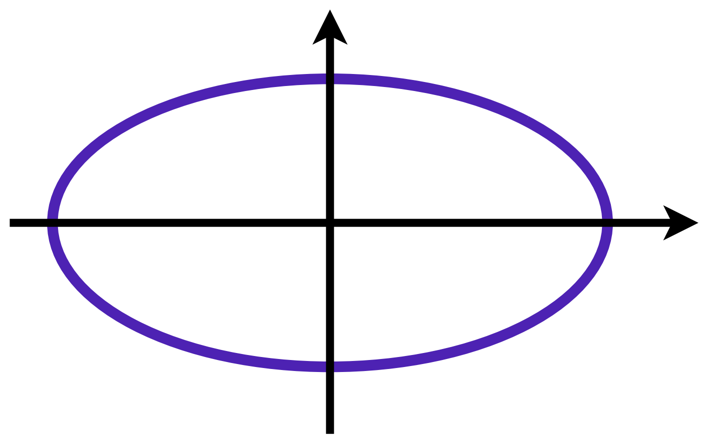
- Motivation

✓ Easy modeling of motions that involve topological change, e.g. shapes splitting or merging

✓ Such operations are difficult to encode with meshes, since they don't "split" or "merge" unless we force them to

Implicit curves and surfaces (aka. Level-Sets)

- Familiar representations address *some* of these demands:
  - ✓ e.g. Analytic equations
    - ➡ For an ellipsis:



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

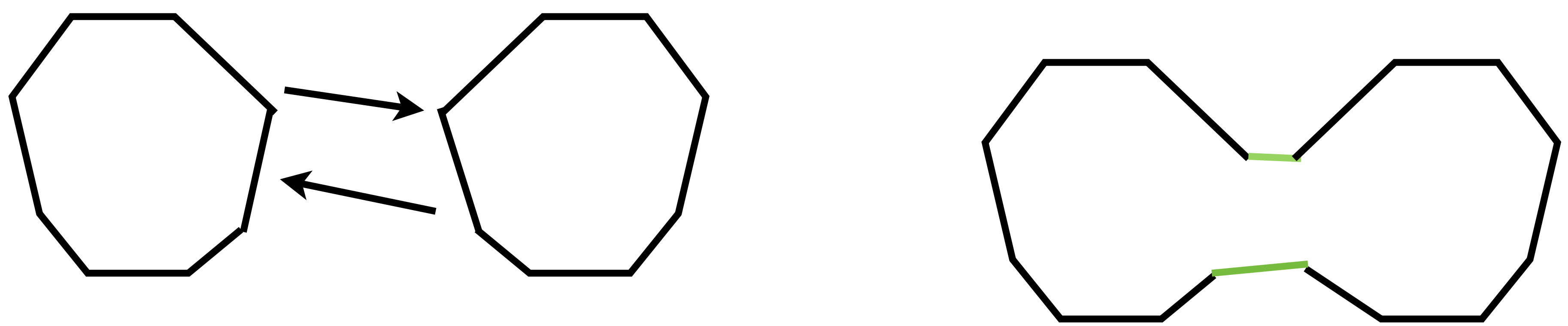
    - ➡ Easy inside/outside tests

$$\frac{x_*^2}{a^2} + \frac{y_*^2}{b^2} < 1 \Leftrightarrow (x_*, y_*) \text{ is inside}$$

Implicit curves and surfaces (aka. Level-Sets)

- Familiar representations address *some* of these demands:

  ✓ Describe a closed region via its boundary; split and reconnect when necessary



➡ This may be tractable in isolated cases, but very cumbersome and impractical for more complicated cases, and with 3-dimensional surfaces

The level-set idea

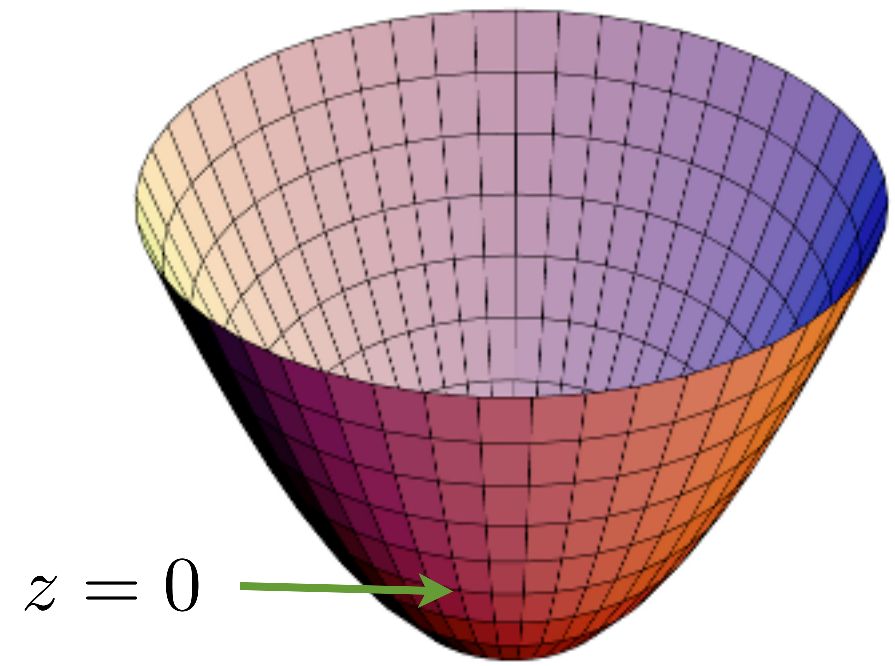
- Represent a curve in 2D (or, a surface in 3D) as the zero isocontour of a (continuous) function, i.e.

$$\mathcal{C} = \{(x.y) \in \mathbf{R}^2 : \phi(x,y) = 0\}$$

e.g.

$$\text{circle } x^2 + y^2 = R^2 \equiv \{(x,y) : \phi(x,y) = 0\}$$

$$where\ \phi(x,y) = x^2 + y^2 - R^2$$



The level-set idea

- This representation may seem redundant (we store information everywhere, just to capture a curve), but it conveys important benefits:

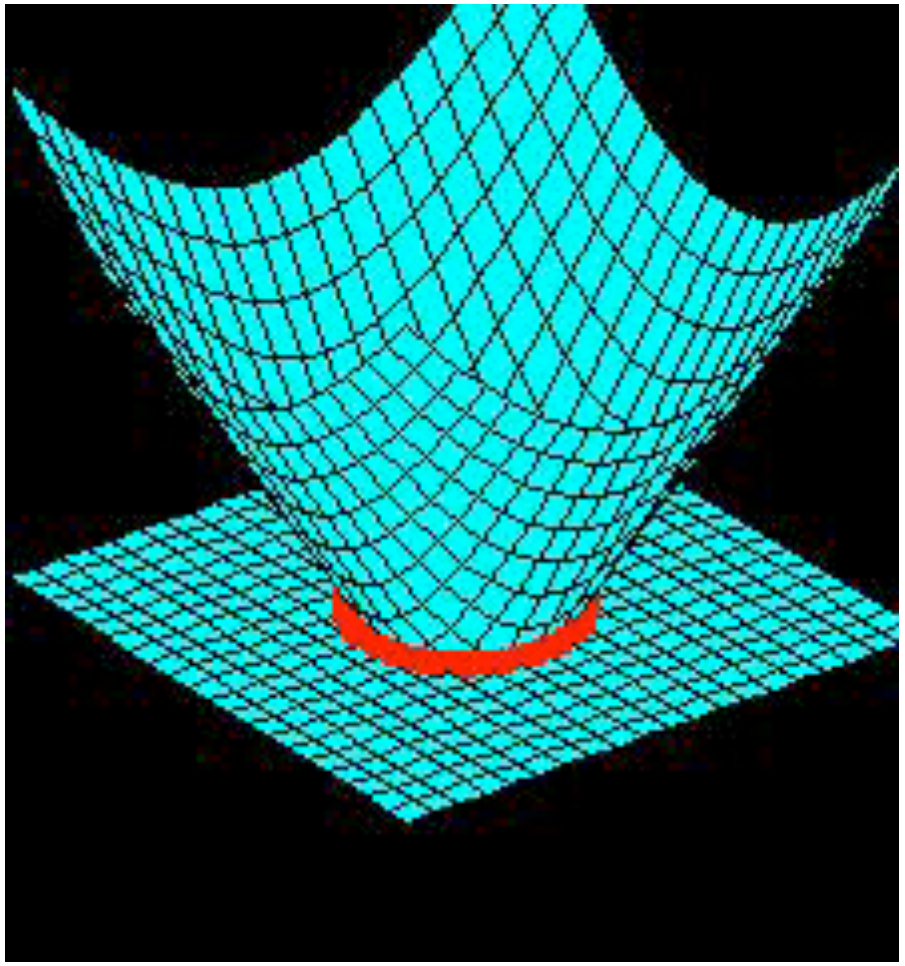
➡ Containment queries

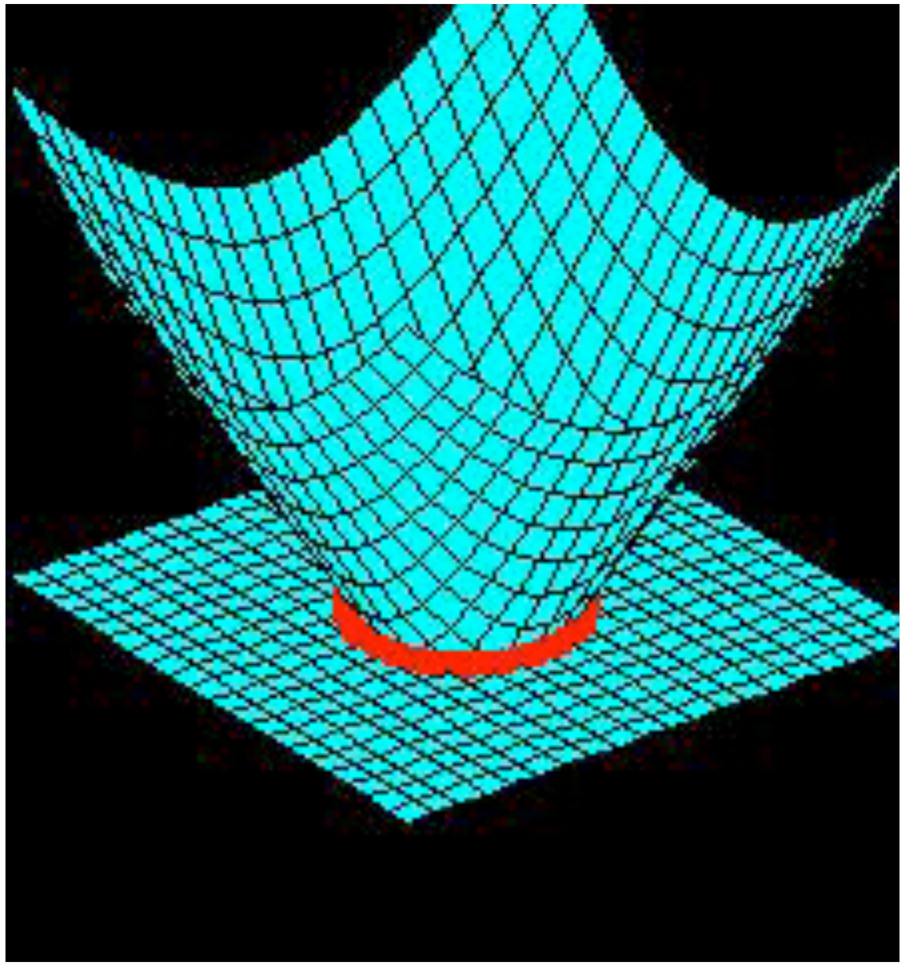
$$\text{Is } (x_*, y_*) \text{ inside } \mathcal{C}? \Leftrightarrow \phi(x_*, y_*) < 0$$

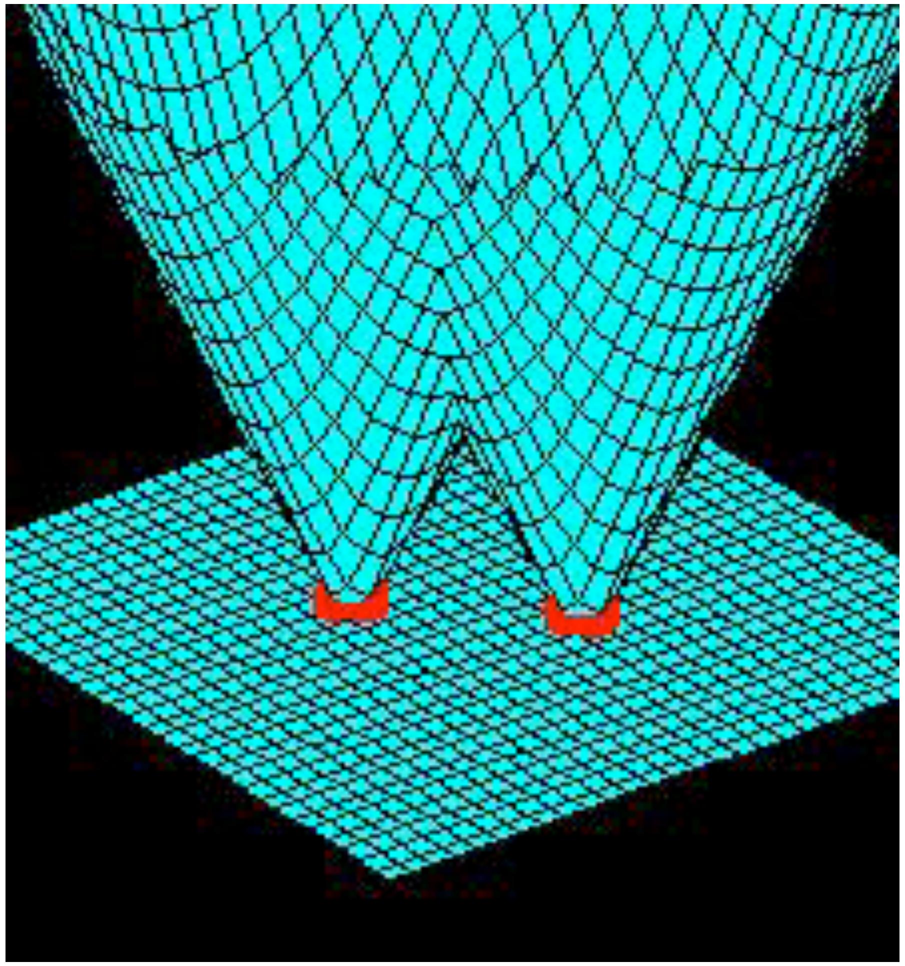
➡ Composability

$$\left.\begin{array}{l} \phi_1(x,y) \text{ encodes } \Omega_1 \\ \phi_2(x,y) \text{ encodes } \Omega_2 \end{array}\right\} \Rightarrow \quad \begin{array}{l} \max(\phi_1, \phi_2) \text{ encodes } \Omega_1 \cap \Omega_2 \\ \max(\phi_1, \phi_2) \text{ encodes } \Omega_1 \cup \Omega_2 \end{array}$$

➡ We model both shape & topology change by simply varying the level set function







Levelset construction

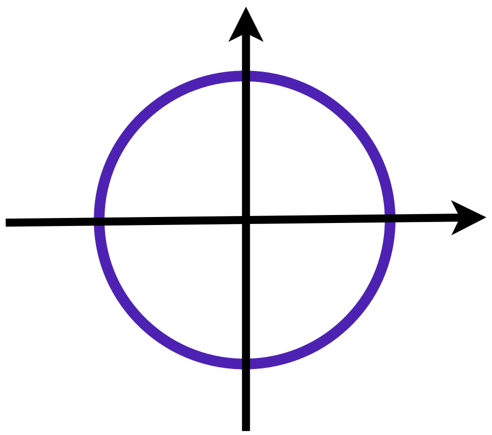
- By the initial definition, there are many level set functions that encode the same shape

$$\left.\begin{aligned} \phi(x,y) &= x^2+y^2-R^2 \\ \phi(x,y) &= \sqrt{x^2+y^2}-R \\ \phi(x,y) &= e^{x^2+y^2}-e^{R^2} \end{aligned}\right\} \text{ all encode the circle } x^2+y^2=R^2$$

- A specific systematic construction process: *Signed distance functions*

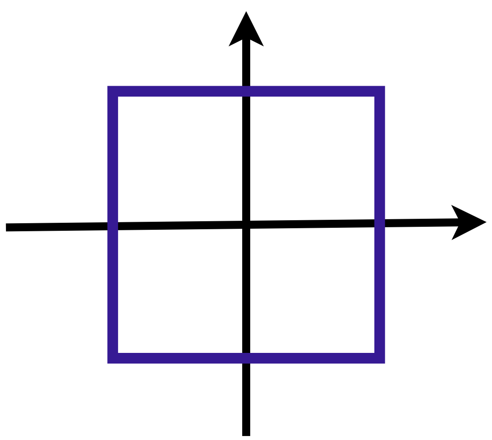
$$\left.\begin{array}{c} \phi(x,y)<0, \text{ if } (x,y) \text{ is inside } \mathcal{C} \\ \phi(x,y)>0, \text{ if } (x,y) \text{ is outside } \mathcal{C} \\ \phi(x,y)=0, \text{ if } (x,y) \text{ is on } \mathcal{C} \end{array}\right\} \text{ and } |\phi(x,y)| = \text{distance of } (x,y) \text{ from } \mathcal{C}$$

Examples



$$\phi(x, y) = \sqrt{x^2 + y^2} - R$$

$$\phi(x, y) = \max\{x - R, -x - R, y - R, -r - R\}$$



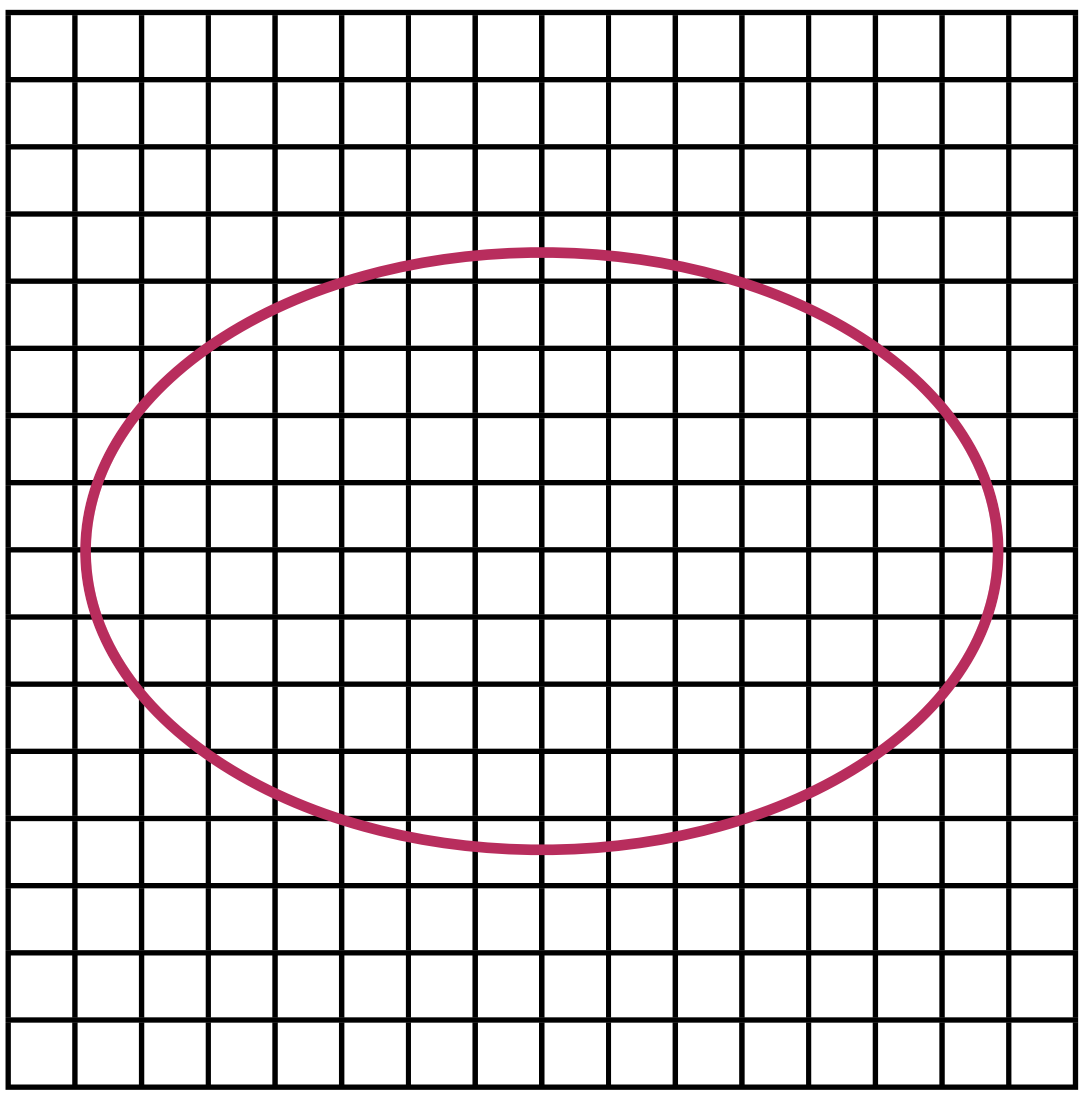
Properties

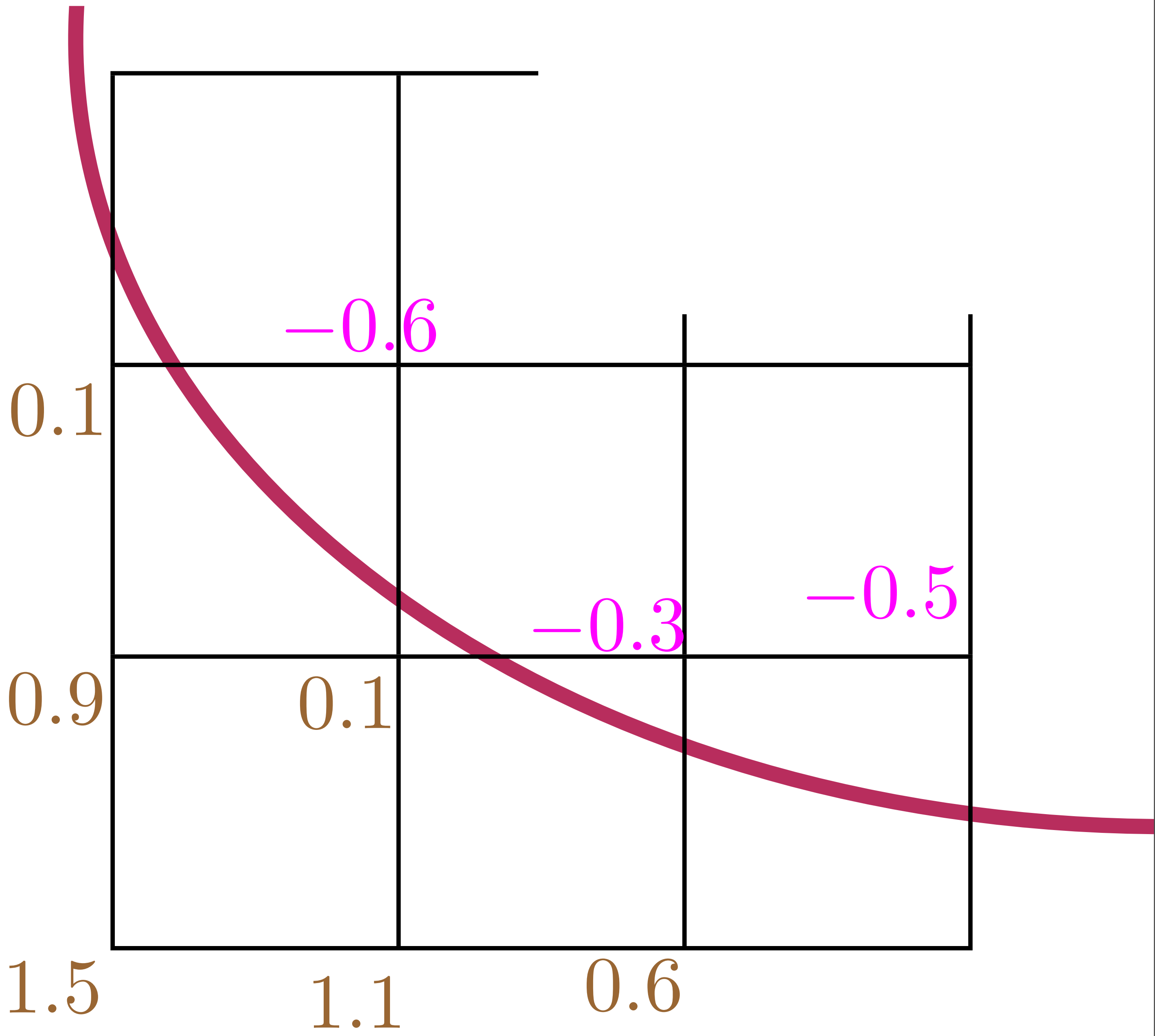
- We can offset the surface by a fixed distance D, by simply adding/subtracting D from the levelset function
- Proximity queries can be answered in O(1) time
  - ✓ e.g. *Is point (x*,y*) within 0.1 units of the surface?*
- The level set gradient is a unit normal, parallel to the direction of the closest point on the surface
- We can project to the surface in O(1) time

$$(x_c, y_c) = (x, y) - \phi(x, y) \cdot \nabla\phi(x, y)$$

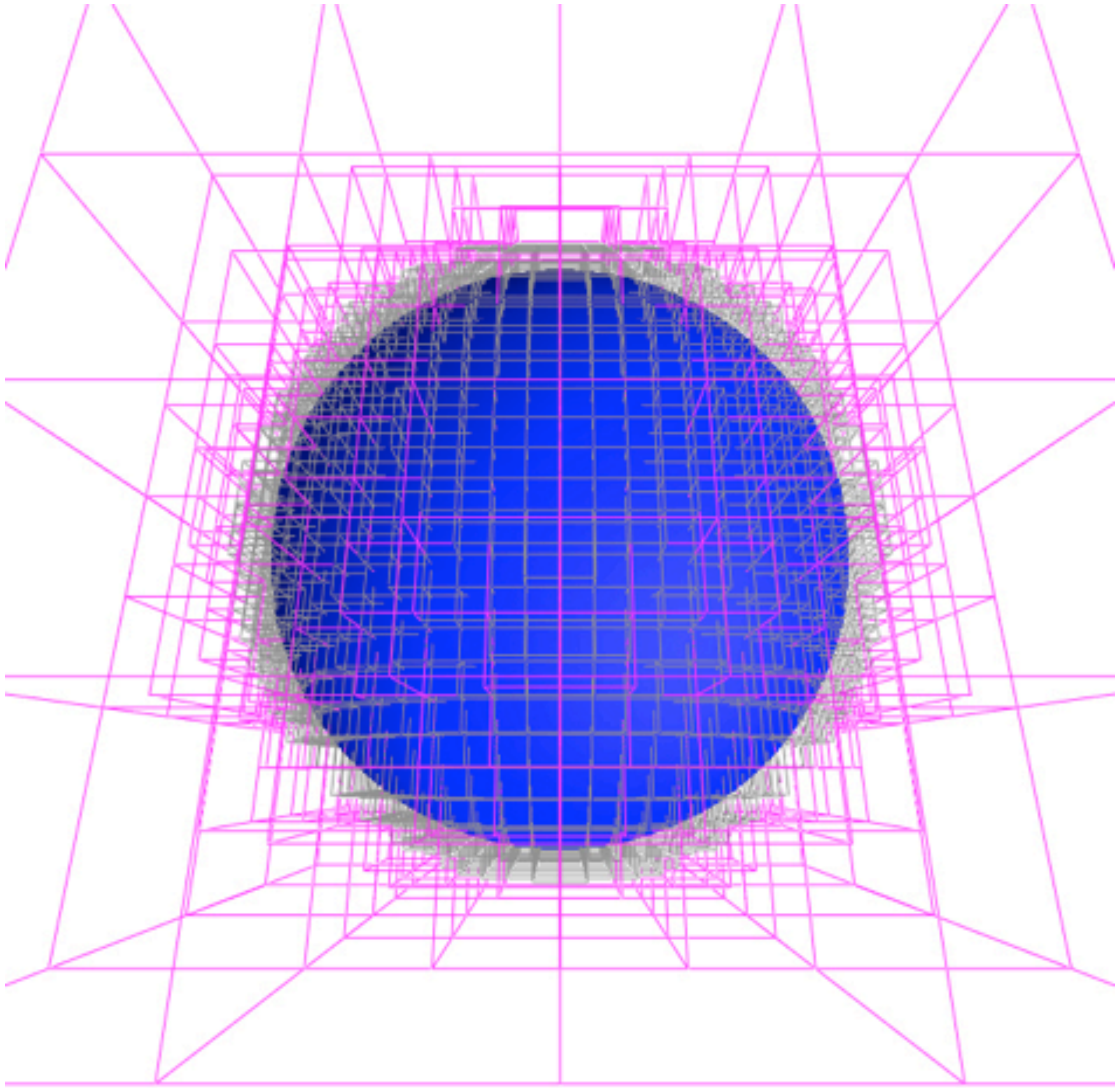
- SDFs are composable over unions/intersections of implicit domains

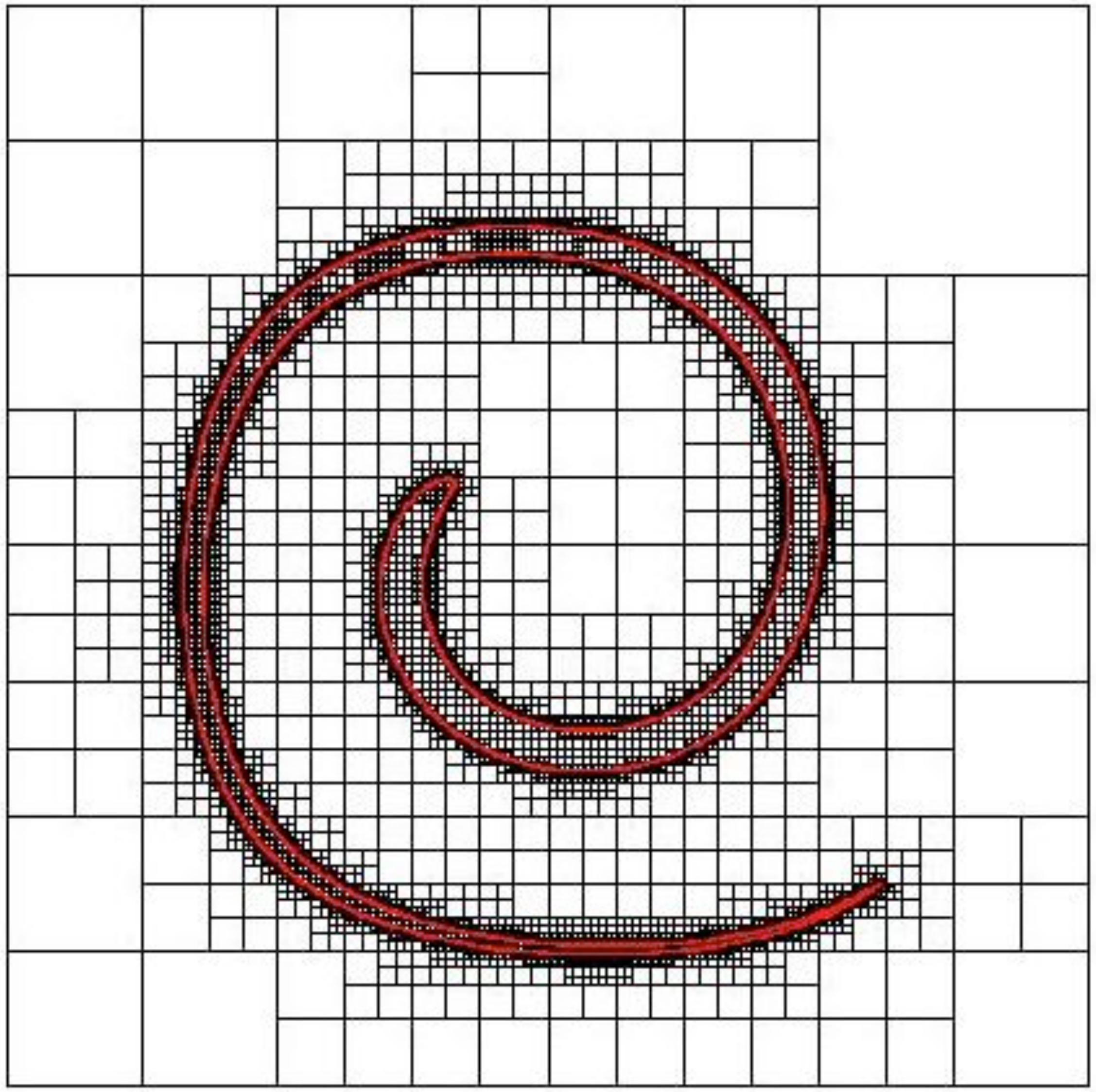
Implementation





Implementation (with adaptivity)





Discrete representations of geometric objects: Features, data structures and adequacy for dynamic simulation: Part II : Levelsets & implicit surfaces