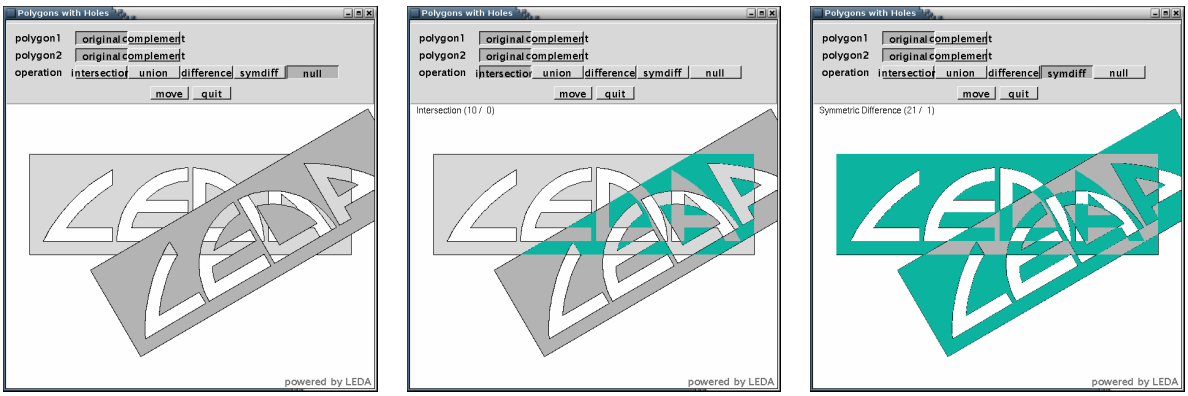

Figure 1: The left part shows two polygonal regions with holes (in light and dark grey). The middle part shows the intersection of these regions and the right part shows the symmetric difference. The figure was produced with the LEDA demo polygon_logo [4, 3].

Lecture 1
Introduction

1 Geometric Computing

Geometric computing refers to computation with geometric objects such as points, lines, hyperplanes, disks, curves, surfaces, and solids.. These objects live in an ambient space. In this book, ambient space will be mainly two- and three-dimensional Euclidean space. Geometric computing is ubiquitous. We illustrate its richness by way of examples. add

Computer-aided Design: Computer-aided design is about the construction of geometric objects. Starting from a ground set of geometric objects, e.g., half-planes, circles, ellipsoids in the plane or cubes, spheres, cylinders, tori, one constructs complex shapes by applying geometric operations to previously constructed objects. Figures 1 and 2 show examples in two and three dimensions, respectively. Figure **??** shows a more complex example.

Robotics: A central task of robotics is the planning of collission-free paths. Consider a simple situation; the goal is to move a disk-like robot amongst polygonal obstacles in the plane, see Figure **??**. The Voronoi diagram of the obstacles is an appropriate data structure for the task. It consists of all points of maximal clearance from the obstacles; a disk grown at a point of the Voronoi diagram hits two or more obstacles simultaneously. The diagram represents paths for maximal safety. In order to move a disk from a point A to a point B, we first move it from A to a point on the Voronoi diagram, then along the Voronoi diagram, and finally from the Voronoi diagram to B.

Graphics: A 3D scanner is a device that analyzes a real-world object or environment to collect data on its shape and possibly its appearance (i.e. color). In its simplest form it returns a set of points on the surface

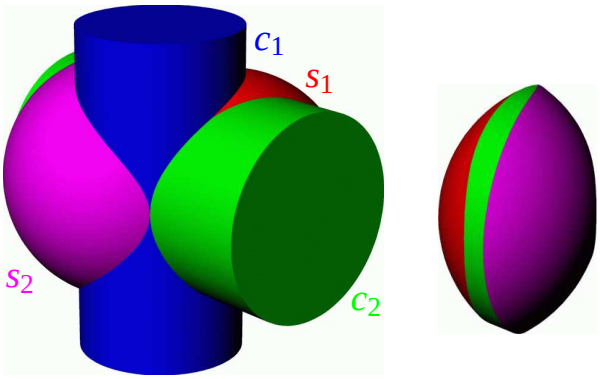

Figure 2: The left part shows four solids: two cylinders and two spheres. The right part shows their intersection. The surface of the intersection composed of faces (surface patches stemming from one of the solids), vertices (intersection curves between two input solids) and vertices (points in common to three or more input solids). The picture was produced with the CAD-software Rhino3D.

of the object, see Figure 4. The geometric computing task is then to construct a digital three dimensional model of the object from the collected data. The task arises in the production of movies and video games. Other common applications of this technology include industrial design, orthotics and prosthetics, reverse engineering and prototyping, quality control/inspection and documentation of cultural artifacts.

Linear Programming: Linear programming is concerned with the optimization (maximization or minimization) of a linear function subject to linear constraints:

$$\text{maximize} \quad c^T x \quad \text{subject to} \quad Ax \leq b,$$

where x is a vector of n variables, $c \in \mathbb{R}^n$ defines the objective function, $A \in \mathbb{R}^{m \times n}$ is a $m \times n$ real matrix and $b \in \mathbb{R}^m$ is a real vector. Each row a_i of A and the corresponding entry b_i of b defines a linear inequality $a_i x \leq b$. Geometrically, the set of x satisfying this inequality form a halfspace in $\mathbb{R}^n$. The set of x satisfying all constraints $Ax \leq b$ is the intersection of halfspaces, i.e., a convex polyhedron P in $\mathbb{R}^n$. Figure 5 shows an example. The aim of linear programming is to find a point $x \in P$ that maximizes $c^T x$. The maximum is attained at a vertex of P that is maximal in direction c.

Mathematics: Algebraic curves and algebraic surfaces are an important topic in mathematics. An algebraic curve is the zero set of a polynomial $p(x,y)$ in two variables and an algebraic surface is the zero set of a polynomial $p(x,y,z)$ in three variables. In applications of algebraic curves and surfaces, it is important to visualize them. Figure 6 shows some examples.

More Examples: continue definition with pictures, give examples, examples should come form computational geometry, but also from fields outside CS, e.g.,

- medicine: reconstruction of artery system in brain from NMR-images
- searching for patterns in astronomy

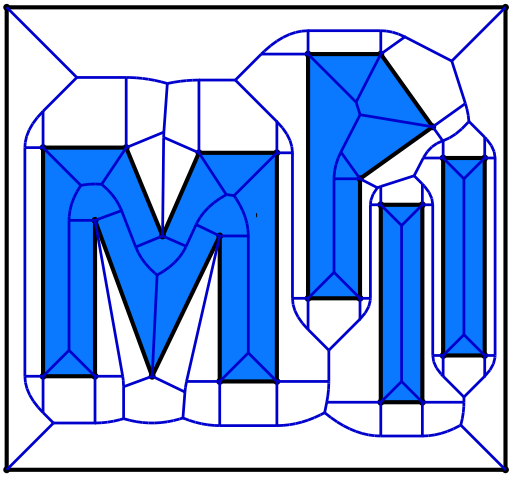

Figure 3: Robot motion planning: The figure shows four polygons (the letters M, P, I, and I) enclosed in a square frame. The space between the polygons and the space inside the polygon in partitioned into cells by the Voronoi diagram of the polygons. Imagine to grow a disk centered in an arbitrary point of the plane. In general, the first collission of the growing disk with one of the polygons will be with a single polygon. The Voronoi diagram consists of all points, where this first collission involves two or more polygons. The Voronoi diagram (also called Medial axes) comprises the points of maximum clearance from the disks. The figure was created by Michael Seel [5].

- have a look at Danny Halperin's page: he has nice examples with pictures. Also he taught a course on applied computational geometry.
- GIS: map overlay, map simplification, map labelling,
- examples from the book of Overmars

2 Preview of the Course

Now that we have developed an intuition for geometric computations, we are ready for an overview of the course.

Lecture II: We will start with a simple geometric problem, the computation of the convex hull of a finite set of points in the plane. We will see several algorithms for solving the problem based on different computational paradighm: incremental computation, sweep, and divide-and-conquer. We will formulate the algorithms in terms of geometric predicates. The primitive required for the convex hull problem is the orientation predicate for three points. Given three points p, q, and r in the plane, the predicate tells whether the points form a left turn, are collinear, or form a right turn; see Figure 7 for an illustration. The triple (p,q,r) is a left turn if $p \neq q$ and r lies to the left of the line passing through p and q and oriented from p to q.

t

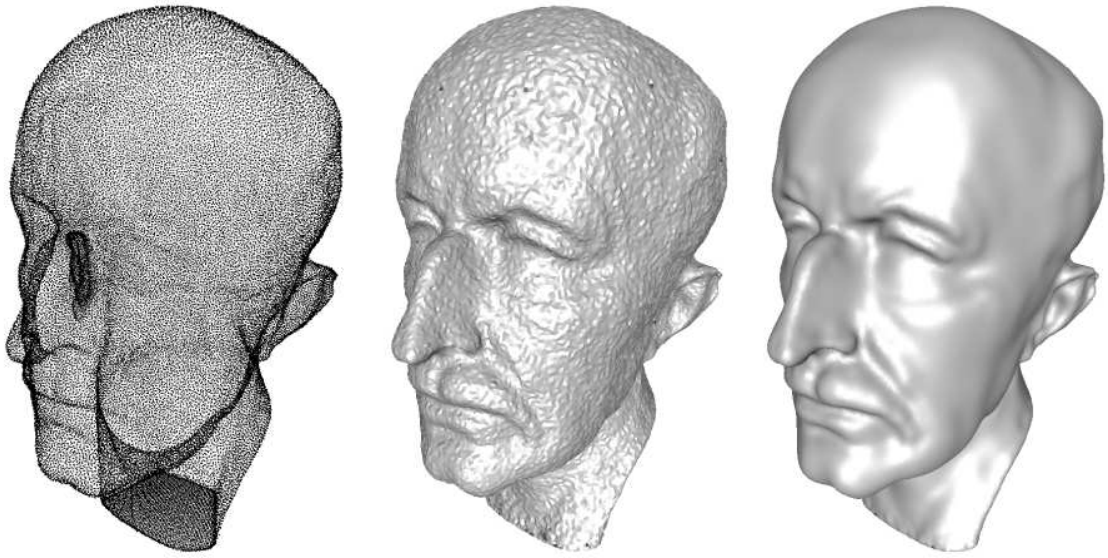

Figure 4: The left part of the picture shows a point cloud obtained from a 3D-scan of a bust of Max Planck. The middle part and right part show reconstructions of the object (non-smoothed and smoothed). The reconstruction is by Tamal Dey, University of Ohio.

Lecture III: Points are usually represented by their Euclidean coordinates. We derive an analytical formula that expresses the orientation of three points in terms of their coordinates. We will see that

$$\mathsf{Orientation}(p,q,r) = \mathrm{sign}(\det \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix}),$$

where p_x and p_y are the x- and y-coordinate of p, respectively. The sign is $+1$ if (p,q,r) form a left turn, is 0 if they are collinear, and is -1 if they form a right turn.

Point coordinates are real numbers as are the parameters defining other geometric objects, e.g., the coordinates of the center and the radius of a disk. Therefore the natural model of computation for geometric computing is the *Real-RAM*. It is a random access machine with the additional capability of handling real numbers. Of course, the operations on real numbers follow the laws of mathematics. The Real-RAM model is also used successfully in numerical analysis.

Real computers do not come with real arithmetic. They provide only floating point arithmetic and bounded integer arithmetic. We will study the effect of floating point arithmetic on geometry. We will first see the effect on the orientation predicate (see Figure **??**) and then the effect on our convex hull algorithm (see Figure **??**). The former effect will be surprising, the latter effect disastrous.

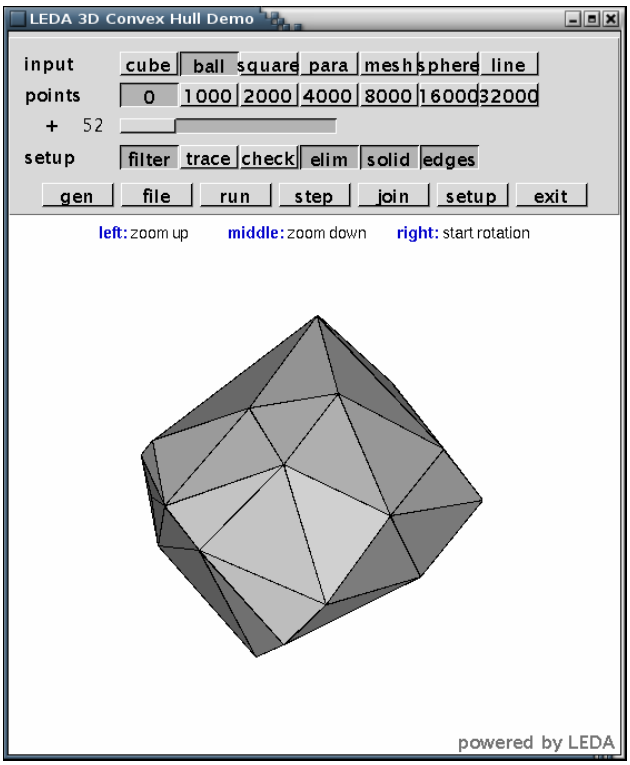

Figure 5: A convex polyhedron in three dimensional space. It was generated as the convex hull of a set of points (using the LEDA demo 3d-hull [4, 3]). Alternatively, it could be constructed as the intersection of the halfspaces corresponding to the faces of the polyhedron. Linear programming finds the extreme vertex in the direction of the objective function.

3 A First Algorithm: Convex Hulls in the Plane

4 Reliable Geometric Computing

mention LEDA and CGAL
examples of non-robustness:

- Chee's write-up
- my examples from Certifying Algorithms paper We give some examples of failures in algorithmic software.
  - Intersection of Solids:
    Rhino3d (a CAD systems) fails to compute correct intersection of two cylinders and two spheres
  - Linear Programming Solvers: CPLEX (a linear programming solver) fails on benchmark problem *etamacro*. Review [2].
  - Mathematica 4.2 (a mathematics systems) fails to solve a small integer linear program

Figure 6: A figure of an algebraic curve (from Pavel's gallery, a close-up view of a singularity, a figure of a triangulated surface. Show the equations, either in the text or in the caption.

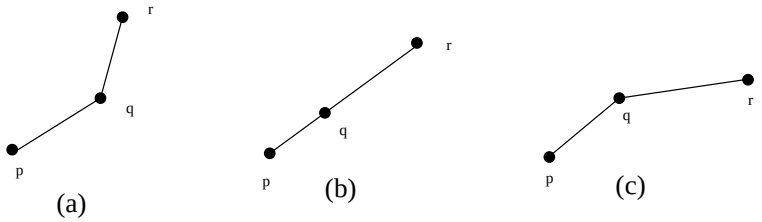

Figure 7: (a) shows a left turn, (b) shows collinear points, and (c) shows a right turn.

```
In[1] := ConstrainedMin[ x , {x==1,x==2} , {x} ]
Out[1] = {2, {x->2}}

In[1] := ConstrainedMax[ x , {x==1,x==2} , {x} ]
ConstrainedMax::"lpsub":  "The problem is unbounded."
Out[2] = {Infinity, {x -> Indeterminate}}
```

– point in line

also discuss here that we want to compute non-continuous functions.

Pentium Bug: A version of the Pentium chip contained an error in the division hardware [1].

This book is about efficient reliable geometric computation.

A general discussion of reliability, examples of non-reliable computation, this is not restricted to geometric computation.

examples of geometric computation by pictures. Define the terms in the title:

- geometric computation = computation with geometric objects, define in words and in pictures. Give examples of geometric computing. Should come from a variety of sources, also outside CS, e.g., graphics, medicine, astronomy,
- reliable = program does what it claims to do, program comes with a guarantee. programming with contracts

  Examples are:

  – program computes a relation $\psi(x,y)$, i.e., for every input x it delivers a y with $\psi(x,y)$
  – program computes an approximate solution in a precise sense.
  – program computes an approximate solution in an intuitive sense.
  – program never crashes
- efficient comes in two flavors:

- theoretical efficiency: low asymptotic running time
- practical efficiency: program is useful, can compete with alternatives,

References

[1] M. Blum and H. Wasserman. Reflections on the pentium division bug. *IEEE Transaction on Computing*, 45(4):385–393, 1996.

[2] M. Dhiflaoui, S. Funke, C. Kwappik, K. Mehlhorn, M. Seel, E. Schömer, R. Schulte, and D. Weber. Certifying and Repairing Solutions to Large LPs, How Good are LP-solvers?. In *SODA*, pages 255–256, 2003.

[3] LEDA (Library of Efficient Data Types and Algorithms). `www.algorithmic-solutions.com`.

[4] K. Mehlhorn and S. Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[5] M. Seel. Eine Implementierung abstrakter Voronoidiagramme. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1994.