Homework 8

Math 282 Computational Geometry
due 5:00pm on Thursday, April 15

Solve the following problems from the textbook, and write your solutions clearly and neatly. Make sure to explain your reasoning and provide mathematical details that support your answers. For a few tips on writing solutions, see this helpful guide for mathematical writing.

These exercises are for everyone, regardless of whether or not you are taking this course for CS elective credit.

You may write or type your solutions electronically, or write them on paper and scan/photograph them. If you photograph your papers, please use a scanning app to produce a single PDF file containing your solutions. Upload your written solutions (and your code/output if relevant) to the Homework 8 assignment on Moodle.

1. Exercise 4.31

2. Exercise 5.11

3. Exercise 5.12

4. Exercise 5.15

5. Exercise 5.16

6. Exercise 5.17

7. Do one of the following two exercises, which are related to Section 4.4 in the text.

(a) Let R and B be two sets of 2D points, colored red and blue, respectively. We want to determine whether there exists a circle that encloses all the red points and excludes all the blue points. If such a circle exists, we want to find one.

Use the paraboloid $z = x^2 + y^2$ to translate this to a problem into a 3D problem that is easier in the sense that involves finding a planes rather than finding a circle. State the 3D problem clearly. Explain a solution to the 3D problem gives a circle that solves the 2D problem.

Then design an algorithm for solving the 3D problem. A brute-force inefficient algorithm suffices, as long as it really would always work. Describe your algorithm at a high level, giving enough detail so that your algorithm is unambiguous. You don't need to implement it in code (though you can if you want).

(b) Write code that computes the convex hull of a set of 3D points, then separates the lower hull from the upper hull and displays each separately. You may do this in any programming language you like, and you may use built-in functions (such as Mathematica's ConvexHullMesh). To decide whether a triangle is on the lower or upper hull, you could compute an outward-pointing normal vector and examine its z-coordinate.