New Courses

Computational Topology

Çağatay Demiralp, PhD student and TA for CS195H

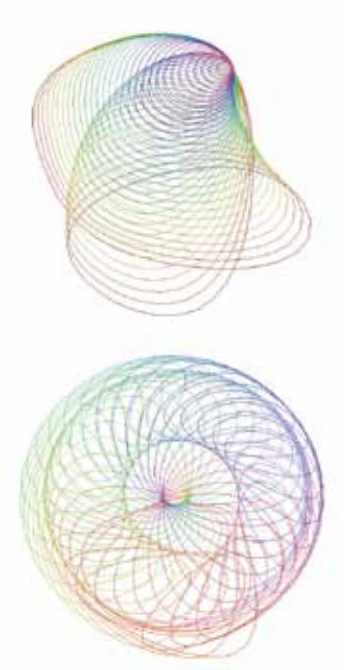

Figure 1: Screenshots from student projects plotting regular homotopy steps between two smooth polygonal curves

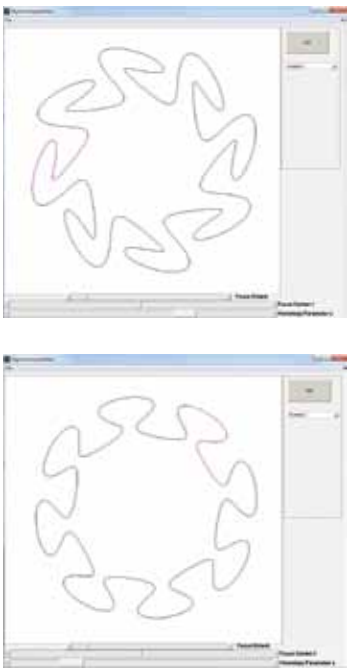

Figure 2: Snapshots from the Thurston corrugations demo used in CS195H.

CS195H - Computational Topology is a new addition to the department's course offerings by Spike (a.k.a. John F. Hughes) and taught first time this spring (Spring 2011). As its name suggests, the course aims to introduce students to various algorithmic problems that arise in the computational study of topology.

Compared to other branches of mathematics, such as geometry or analysis (e.g., calculus), computer science students are often less familiar with topology. One reason for this is that topology is a relatively new field. Although its roots can be traced back to Euler and Gauss (well, that can be said for almost every other branch of mathematics), topology is primarily a product of the 20th century mathematics and was pioneered by Poincaré. So, what is topology about? It is best understood in a comparison with geometry. In the Euclidean geometry (one of many possible geometries), one studies the congruence (or equivalence) of spaces (e.g., curves and surfaces) under rigid motions, which preserves lengths and angles. On the other hand, in topology we study the congruence of spaces under continuous deformations. This means that we deem changes in lengths and angles to be unimportant. For example, consider the two font curves "**S**" and "**I**"; although they are not congruent in the Euclidean geometry, these two curves considered "the same" in topology (here is a continuous deformation that projects the curve **S** onto **I**: *Pull the endpoints of the curve* **S** *until it is straightened and then shrink it from both ends to match with* **I**). Intuitively, this added "flexibility" in the transformation under which the congruence is considered should make it easier for different spaces to be congruent. So, in topology, one should expect a smaller number of equivalence classes of spaces.

Both topology and computational topology are very active research fields. For example, Grigori Perelman's proof of the Poincaré conjecture and the drama surrounding the subsequent events are very recent memories. Similarly, computational topology has recently started finding increasing number of applications in a wide range of areas such as low-level vision, global (or qualitative) data analysis, and sensor networks.

Like topology, computational topology is a large and diverse field and the course syllabus is just a selection from it, reflecting, in part, Spike's interests. Here are some examples from the course projects (See the course website http://www.cs.brown.edu/courses/csci1950-h/ for further details):

- *Homotopy:* Show that every grid loop (a loop whose vertices are on the integer lattice, and whose edges are all vertical or horizontal) is homotopic to a constant grid loop by writing a program that takes a grid loop as input and produces as output a sequence of grid-homotopy steps. (Grid homotopies are made of discrete steps that model continuous deformations). Modify this program to take any grid loop in the punctured plane (a plane with a missing point) and deform it into one of the "standard" loops (or the constant loop if it doesn't wind around the missing point).
- *Whitney-Graustein Theorem:* Write a program that computes the turning number of a polygon. Given a polygon of n vertices and turning number k, construct a regular homotopy from the given polygon to the standard polygon with n vertices and a turning number k. Figure 1 shows screenshots from student projects plotting regular homotopy steps between two smooth polynomial curves.
- *Immersions:* Write a program that lets the user click a polygon, and then determines whether the polygon bounds an immersed surface. Enhance your program so that if the polygon bounds one or more surfaces, you report the turning number of the polygon and the Euler characteristic of each surface that it bounds.
- *Simplicial Homology:* Write a program that computes the bases for the groups of cycles that do not bound for a given triangulated surface (e.g., torus, Klein bottle, RP^2, etc.) or a higher-dimensional simplicial complex with coefficients in Z/2Z (binary numbers) or Z (integers).

The course is primarily aimed at Junior/Senior Math/CS concentrators who have taken a 100-level math course that involves some topology (e.g., MA106, MA141, and MA126) and who know how to program, and preferably have some experience with Matlab. However, in my experience as a TA so far, it is also a great way to learn topology for those who have not taken any related math courses because concepts are

introduced very intuitively and programming projects force students to think about the underlying ideas and theorems at a depth that these ideas and theorems often demand.

Spike is very excited about the course and has already written more demo code than we the TAs (Andrew Furnas, a stellar math concentrator with accumulated accomplishments such as Goldwater and Marshall Scholarships, is the HTA for the course). Figure1 shows snapshots from the interface for his Matlab code to demonstrate regular homotopy via *Thurston corrugations*. Well, you will need to take the course to learn what that means.

CS2580

by Carleton Coffrin, PhD Student & TA for CS2580

Back in the summer of 2007 I was working out of my apartment in NYC and coding web sites for anyone willing to pay. Of course there were advantages to freelancing: sleeping in every day, working in my pajamas, having no boss, but the work just wasn't exciting for me. I spent my afternoons debugging cross-browser css compatibilities and dreamed all night of writing algorithms to "change the world"! (I will admit that just one year out of undergrad, my youthful optimism was as strong as it ever had been.) I had heard whispers in my last year of undergrad of optimization tools, and the prospects of using this technology for solving real-word problems were very exciting. So I decided to return to school and master these optimization technologies. I imagined that my time in academia would focus on toy problems such as sudoku or magic squares, but little did I know that I would be solving real-world problems as soon as I arrived at Brown.

I joined Pascal's lab (Brown's Optimization Laboratory) in the fall of 2008, and before the new student orientation had finished Russell Bent ['05] had already given an exciting talk about a series of disaster recovery problems that his division faces at Los Alamos National Laboratory. This was the beginning of a collaboration between Brown's Optimization Laboratory and Los Alamos's Energy and Infrastructure Analysis division that is still thriving today.

In my first year at the Optimization Laboratory we successfully developed an algorithm for a potable water disaster recovery problem. I was able to spend the summer continuing this work on site at Los Alamos National Laboratory, and integrated it into their fast response pipeline. Each time a category three or higher hurricane is projected to make land fall on US soil, the fast response pipeline is executed and cutting-edge algorithms are used to provided decision support to US government officials. It seems that my dreams of "changing the world" with computer science were coming true within just one year of starting my graduate studies. "Changing the world" maybe a bit of an exaggeration, but I am still excited help save lives during natural disasters.

Results from the weekly class tournament where green boxes highlight the top performers. The competition is fierce!

Providing decision support to the US government is just the beginning. We also must share our knowledge with the broader community. We often do this through academic publications, but we have also chosen to disseminate our research directly to the Brown community through Pascal's graduate course, "Solving Hard Problems." Solving Hard Problems is not new to the course catalogue and the weekly class tournaments remain, but we have adjusted the curriculum to focus on problems in the field of humanitarian logistics. We have added two more projects to the course and changed the others to be focused on decision-support applications. Some of the new projects include: "Supply Chain," where a transportation network must be optimized to deliver relief supplies; "Green Zone," where security forces must be efficiently deployed to cover an urban area; and "Special Operations," where medical supplies must be packaged efficiently while minimizing the risk of contamination. So far the response to the new curriculum has been positive, and we hope that some of the students will have the opportunity to take their experience outside the classroom and use it to solve real-world problems. In fact, two students, Nell Elliott and Ben Simon, are already working in the Optimization Laboratory on disaster-recovery projects.