

Teaching Statement

I am active in a project involving Georgia Tech, Bryn Mawr College, and Microsoft Research to use personal robots as vehicles for learning introductory computer science. In addition to helping design the hardware platform and being a chief developer of the robot software, I have also had the pleasure of teaching the class. Teaching this freshmen level computer science class has been one of the most rewarding aspects of my graduate school career. I have learned first-hand that although robots may win the attraction battle, it is good instruction that wins the war. Coming up with lessons that convey abstract concepts like recursion and conditional execution using embodied computers like robots is both challenging and rewarding. For instance, recently I gave a lecture on using our robot to create a silent movie. Although presented as a film strip, in reality, the students were learning how to create, traverse, and manipulate lists. As part of this project, I have been the lead organizer of a workshop on educational robotics at the last Robotics: Science and Systems (RSS) conference, and have proposed a tutorial on our approach at the next International Conference on Robotics and Automation (ICRA).

I encourage, whenever possible, open-ended learning – where students pose their own problems as well as solutions. In my experience mentoring undergraduate and masters students in directed studies, allowing students to pick their own problems has lead them to become heavily invested in the learning process. I look forward to teaching computer science majors in classes ranging from introductory courses to higher-level classes as diverse as programming languages, operating systems, distributed systems, robotics and artificial intelligence. I am also excited about the opportunity to teach non-majors about computing, bridging the gap between computer science and other fields, such as philosophy, psychology, and biology.

Teaching Philosophy

“The concepts of science, in all their richness and ambiguity, can be presented without any compromise, without any simplification counting as distortion, in language accessible to all intelligent people.” – Stephen Jay Gould

The preceding quote opened the personal statement of my graduate school application. Yet, almost six years later, it motivates me more than ever. Computer science is as fundamental to modern intellectual inquiry as mathematics, physics, or biology. However, the core principles of computing are sometimes distorted, other times over-simplified and almost always over-complicated. This prevents the concepts of computer science – which are as useful to the statistician as the sculptor – from being conveyed to *all intelligent people*.

Although I believe in emphasizing abstract themes and concepts, contextualizing computer science in terms of the real-world and the larger intellectual atmosphere is imperative. Robotics, embedded computing, and other research areas are excellent sources of motivation and illustration. At the very least, they show that computer science is more than bank databases, Fibonacci numbers, and dining philosophers. At most, they provide a novel view into the inner-workings of the algorithm. By making a robot loop in a square or react to its sensors, a more abstract notion of computing is gained.

As educators we must take care in how we use technology. Any technology – especially cumbersome robots – can detract from our educational goals and have maleffects. Trivial details of the particular technology can be mistaken for core concepts resulting in misplaced effort and intellect. This is much to blame for the distortion and mis-information problems facing computer science. However, as practicing computer scientists, details are something we must deal with. At the end of the day, real programs must be written. The solution to this problem is to teach computing with a wide variety of computing platforms. Platforms that not only stretch our perception of a computer but help us appreciate the abstract notion of universality of computation. Rather than learning the general concepts directly in an ungrounded manner, the students gain a general perspective on computing by working with a variety of specialized platforms. Whether the platforms are robots, web applications, games, bank databases, or the lambda calculus.