Resumé

David Zhao Akeley

UCLA Engineering undergraduate

Majors Computer Science, Mathematics **GPA** 3.745 **Expected Graduation** August 2020

Select Engineering Courses Parallel & Distributed Computing, Advanced Computer Architecture

Select Mathematics Courses Complex Analysis Honors, Algebra Honors

Work Experience

× Sholari LLC – July - September 2019 – Contractor

1. Worked on a Unity 3D game that simulates tumor growth and provides visualizations of tumor response to various treatment options.
2. Implemented line graphs, waterfall (bar) plots, and the user interface for the timeline (graph x-axis).
3. Wrote a multithreaded `C++11` plugin for visualizing tumors & immune system responses as particle clouds, and integrated it with the single-threaded `C#` Unity Engine.

× Stanford Aetherling Project – June - September 2018 – Research Assistant

1. Aetherling currently aims to support automatic parallelization of hardware image pipelines designed using a Haskell intermediate representation.
2. Contributed to Aetherling's functional simulator and worked to remove impediments to parallelizing Aetherling line buffers.[1]
3. Collaborated with David Durst (lead author), Dr. Kayvon Fatahalian, and Dr. Pat Hanrahan.

`https://github.com/David-Durst/aetherlingHaskellIR`

`https://github.com/David-Durst/aetherling`

× MediocrePy – March - June 2017 – Independent Project

1. Created an optimized library for reducing stacks of telescope images to a single image using pixel means or medians and optional outlier rejection (sigma clipping) for noise reduction.
2. Multithreaded C core with AVX vectorization; C and Python (numpy) interface. Decreased runtime (compared to the Python implementation replaced) from hours to milliseconds.
3. Collaborated with Dr. Zheng Cai, UC Santa Cruz Astrophysics.

`https://github.com/akeley98/MediocrePy`

× Tsinghua Astrophysics – July - August 2016 – Summer Intern

1. Designed a library for fitting and plotting standard microlensing event light curves given a set of brightness data for a star.
2. Used Python, C++, SciPy, Matplotlib.
3. Collaborated with Dr. Shude Mao.

× Jide Technology Co. – June - July 2015 – Summer Intern

1. Product testing for RemixOS, an Android derivative with a desktop-like interface.
2. Wrote international marketing & documentation in English.
3. Collaborated with Jason Zheng and Jeff Zhao (International Marketing Manager).

[1] A line buffer device reads in an image as a stream of pixel values and outputs rectangular portions ("windows") of the image.

Other Projects

× WebGL Jelly Cube Project

Simple mass-spring system simulation written with Javascript, WebAssembly, and WebGL 2.0 (for refractive and reflective effects). Earned third place in the UCLA computer graphics class contest, Fall 2017.[2]

https://github.com/akeley98/JellyMcJelloFace https://youtu.be/YwvMSeB6NzU

× DementedIGPU – Linux Nvidia Setup Script

(Unfortunately, this project no longer works due to Bumblebee's end-of-support).

Laptops with Nvidia graphics cards often work unreliably with the GNU/Linux operating system, especially when attempting to switch between high-performance discrete graphics and low-power integrated graphics. I wrote a Python 3 script that automatically installs and configures software needed to provide a (relatively) reliable option at boot time between high- and low-power graphics.[3] I documented the script liberally in order to make it as beginner-friendly as a command line application can be.

https://github.com/akeley98/DementedIGPU

× Myricube – Experimental OpenGL Voxel Renderer (WIP)

This experiment in hybrid raycast/mesh voxel renderering was inspired by my discontent with Minecraft's low render distance, even on high settings.

Conventional voxel renderers such as Minecraft's work by converting voxel models into a mesh of triangles to draw, possibly with some optimizations like hidden face removal or merging coplanar triangles. This uses the GPU for its designed purpose, but as render distance increases, the GPU is bottlenecked by the large number of small triangles generated. Myricube uses conventional mesh rendering for nearby voxels but draws more distant voxels with an alternative raycasting renderer. This renderer subdivides the voxel model into cubic chunks, computes a minimal AABB[4] for each chunk's visible voxels, and draws the AABBs themselves with a raycasting fragment shader that only checks for intersections within the drawn AABB. Dividing the model into AABBs reduces my renderer's memory bandwidth requirements and time wasted raycasting through empty space, compared to a conventional raycasting renderer.

https://github.com/akeley98/myricube

× Proposed gem5 Hardware Simulator Partial Bypassing Patch

This patch is a refactoring of the `C++` code I wrote as part of my UCLA "Advanced Computer Architecture" course project.

In an out-of-order CPU, hardware bypassing allows instructions stalled waiting for a register to start execution as soon as the functional unit (ALU, memory port, etc.) writing to said register completes, without waiting for the actual commitment to register file. This is crucial for performance, so mainline `gem5` simulates complete bypassing between all functional units in the simulated CPU. However, this is not practical to realize in modern superscalar hardware, as bypassing costs grow quadratically with the number of functional units. My patch splits the CPU's functional units into separate pools, with bypassing simulated only between pools. The commit-to-register-file delay is imposed for values communicated between pools.

Although the patch received positive code reviews, it doesn't look like there's enough interest in the feature to actually merge it into mainline `gem5`.

https://gem5-review.googlesource.com/c/public/gem5/+/27767

https://github.com/akeley98/FU-pools/blob/b4d291429edb6aa4988888656b6867ff99591b90/fu-pools.pdf

[2] https://www.facebook.com/vasilescu.alex/posts/10155206917936588

[3] This automation depends on the user using a system with `apt`, `systemd`, and the `GRUB` bootloader. Tested with Ubuntu 18.04.

[4] axis alligned bounding box

UCLA Education – September 2017 - August 2020 (Expected)

First Major Computer Science **Second Major** Mathematics **GPA** 3.745 (April 2020)

	Title (*In Progress*)	Content Notes
EE M16	Digital Systems	Verilog Lab
EE M116C	Computer Systems Architecture	
CS M152A	Digital Design Lab	Verilog Team Project
CS 251A	Advanced Computer Architecture	`gem5` Hardware Sim Project, Graduate Course
CS 35L	Software Construction Lab	POSIX basics (e.g. pthreads, bash)
CS 111	Operating Systems Principles	Focus on POSIX
CS 118	Computer Network Fundamentals	
CS 130	Software Engineering	Java Team Project
CS 131	Programming Languages	
CS 133	Parallel & Distributed Computing	OpenMP, OpenCL, MPI, GPGPU, FPGA
CS M146	Machine Learning	
CS 161	Fundamentals of Artificial Intelligence	
CS 174A	Intro to Computer Graphics	See WebGL Jelly Cube Project
CS 180	Algorithms & Complexity	
CS 181	Formal Languages & Automata	Regex, CFG, Turing Machines, Decidability
Engr 185EW	Art of Engineering Endeavors	Writing Intensive Team Project
Math 110A	Algebra	Ring Theory
Math 110AH	Algebra Honors	Group Theory
Math 110BH	Algebra Honors	Ring Theory, Module Theory
Math 110C	Algebra	Field Theory, Galois Theory
Math 111	Theory of Numbers	Overview of p-adic Numbers
Math 115A	Linear Algebra	
Math 115B	Linear Algebra	
Math 120A	Differential Geometry	
Math 131AH	Analysis Honors	Metric Spaces
Math 131BH	Analysis Honors	Derivation, Riemann Integration
Math 132H	Complex Analysis Honors	
Math 134	Systems of Differential Equations	
Math 142	*Mathematical Modeling*	
Math 167	*Game Theory*	
Math 170A	Probability Theory	

Note: There is no honors equivalent to the Field Theory Course.

West Valley College Education – 2015-2017

GPA 4.0 (upon transferring to UCLA)

Select Courses

	Title	Content Notes
Math 4B	Differential Equations	
Math 19	Discrete Mathematics	
Psych 2	Experimental Psychophysiology	Experiment Design & Paper
Phys 4D	Modern Physics	Relativity