Finalist Reports 2023-2024

Robots on a Mission: Safeguarding Animals from Wildfire Disasters
Truman Middle School
Background Research
Purpose
Data
Wildfire Simulation NETLOGO
Photos
Conclusion

https://supercomputingchallenge.org

Printed by PNM
Compiled by the
Supercomputing Challenge

Cover: **Robots on a Mission: Safeguarding Animals from Wildfire Disasters**

Truman Middle School
Team Members: **Vincent Nuanes, Josue Ochoa, Ayla Alaya**
Sponsor: **Natali Bareto Baca**

Winner in the Technical Poster Competition

New Mexico Supercomputing Challenge
<u>2023 – 2024 Finalist Reports</u>

Table of Contents

Supercomputing Challenge 2023-2024
Simulating Greatness Since 1990
Designed by Team 13, La Cueva High School

Supercomputing Challenge Vision

The Vision of the Supercomputing Challenge is to be a nationally recognized program that promotes computational thinking in science and engineering so that the next generation of high school graduates is better prepared to compete in an information-based economy.

Supercomputing Challenge Mission

The Mission of the Supercomputing Challenge is to teach teams of middle and high school students how to use powerful computers to analyze, model and solve real world problems.

About the Supercomputing Challenge

The Supercomputing Challenge (the Challenge) is an exciting program that offers a truly unique experience to students in our state. The opportunity to work on the most powerful computers in the world is currently available to only a very few students in the entire United States, but in New Mexico, it is just one of the benefits of living in the "Land of Enchantment."

The Challenge is a program encompassing the school year in which teams of students complete science projects using high-performance computers. Each team of up to five students and a sponsoring teacher defines and works on a single computational project of its own choosing. Throughout the program, help and support are given to the teams by their project advisors and the Challenge organizers and sponsors.

The Challenge is open to all interested students in grades 5 through 12 on a nonselective basis. The program has no grade point, class enrollment or computer experience prerequisites. Participants come from public, private, parochial and home-based schools in all areas of New Mexico. The important requirement for participating is a real desire to learn about science and computing.

Challenge teams tackle a range of interesting problems to solve. The most successful projects address a topic that holds great interest for the team. In recent years, ideas for projects have come from Astronomy, Geology, Physics, Ecology, Mathematics, Economics, Sociology, and Computer Science. It is very important that the problem a team chooses is what we call "real world" and not imaginary. A "real world" problem has measurable components. We use the term Computational Science to refer to science problems that we wish to solve and explain using computer models.

Those teams who make significant progress on their projects can enter them in the competition for awards of cash and scholarships. Team plaques are also awarded for: Teamwork, Written

Report, Professional Presentation, Research, Creativity and Innovation, Environmental Modeling, High Performance, Science is Fun and the Judges' Special Award, just to name a few.

The Challenge is offered at minimal cost to the participants or the school district. It is sponsored by a partnership of federal laboratories, universities, and businesses. They provide food and lodging for events such as the kickoff conference during which students and teachers are shown how to use computers, learn programming languages, how to analyze data, write reports and much more.

These sponsors also supply time on the supercomputers and lend equipment to schools that need it. Employees of the sponsoring groups conduct training sessions at workshops and advise teams throughout the year. The Challenge usually culminates with an Expo and Awards Ceremony in the spring at the Los Alamos National Laboratory or in Albuquerque. During the Covid pandemic, three Expo and Awards Ceremonies, as well as two Kickoffs, were held virtually.

History

The New Mexico High School Supercomputing Challenge was conceived in 1990 by former Los Alamos Director Sig Hecker and Tom Thornhill, president of New Mexico Technet Inc., a nonprofit company that in 1985 set up a computer network to link the state's national laboratories, universities, state government and some private companies. Sen. Pete Domenici, and John Rollwagen, then chairman and chief executive officer of Cray Research Inc., added their support.

In 2001, the Adventures in Supercomputing program formerly housed at Sandia National Laboratories and then at the Albuquerque High Performance Computing Center at the University of New Mexico merged with the former New Mexico High School Supercomputing Challenge to become the New Mexico High School Adventures in Supercomputing Challenge.

In 2002, the words "High School" were dropped from the name as middle school teams had been invited to participate in 2000 and had done well.

In the summer of 2005, the name was simplified to the Supercomputing Challenge.

In 2007, the Challenge began collaborating with the middle school Project GUTS, (Growing Up Thinking Scientifically), an NSF grant housed at the Santa Fe Institute.

In 2013, the Challenge began collaborating with New Mexico Computer Science for All, an NSF funded program based at the Santa Fe Institute that offers a comprehensive teacher professional development program in Computer Science including a University of New Mexico Computer Science course for teachers.

2023—2024 Supercomputing Challenge Awards

34th Annual New Mexico Supercomputing Challenge winners

FIRST PLACE

TEAM: Los Alamos High School

Magnetic Reconnection

TEAM MEMBERS: Tate D. Plohr

MENTOR: Mark Peterson

WELL DONE!

Supercomputing Challenge

SECOND PLACE

TEAM: Albuquerque Academy

Water Runoff and Diversion Simulation

TEAM MEMBERS: Harrison Schiek

SPONSOR: Jay Garcia

GOOD WORK!!

Supercomputing Challenge

THIRD PLACE

TEAM: Santa Fe Preparatory School

Computational Hydrodynamic Analysis for Speed Maximization (CHASM)

TEAM MEMBERS: Luke Rand, Greta Swanson, Nandita Ganesan

SPONSOR: Jocelyn Comstock

CONGRATULATIONS!

Supercomputing Challenge

Supercomputing Challenge students across the state have completed their computational projects on topics such as cancer, astrophysics, fire science, and environmental issues. This year the Supercomputing Challenge celebrated their 34th annual Expo and Awards Ceremony on April 29 and 30, 2024 which featured student final presentations, judging and tours held at the Los Alamos National Laboratory and an award ceremony held in Los Alamos.

In this program, middle school and high school students are mentored by a community of volunteer scientists, computer programmers and professors. Several alumni also serve as volunteers. The Supercomputing Challenge partners include the New Mexico Consortium, Los Alamos National Laboratory, Sandia National Laboratory, Public Service Company of New Mexico, Bigbyte, Microsoft, N3B, Westwind, Holmans, NMTIE, Simtable/Redfish Group and most New Mexico colleges and universities. A complete list of sponsors and supporters of the Challenge is on the website at https://supercomputingchallenge.org/23-24/sponsors.

“I am always impressed with the students in our state. We are so proud to be able to showcase their abilities,” said David Kratzer, the executive director of the Supercomputing Challenge.

By participating in the Challenge, students learn to be prepared and successful in any career or college. They learn to be persistent by practicing their computer programming skills, completing research, and meeting deadlines to cross the finish line. The Challenge likes to refer to itself as an academic marathon, and these students should be recognized as critical thinkers, communicators, collaborators, and computer scientists.

Scholarships worth $17,500 were awarded to participating students. Many other awards were distributed ranging from $50 per team member for being finalists in the academic marathon to team prizes of up to $1000 for 1st and additional prizes for other categories such as teamwork, technical writing, programming ability, and community impact. Random drawings were held for $50 door prizes.

The Supercomputing Challenge is open to New Mexico middle and high-school students, including home-schooled students. Students work in teams and follow their own interests to choose a topic to computationally model. New students interested in becoming involved can make plans to start the next academic year, by contacting consult@supercomputingchallenge.org

Los Alamos NATIONAL LABORATORY
Sandia National Laboratories
2023-2024 Finalist Team Supercomputing Challenge
Computational Modeling
Science
Math
S

Finalist teams receive banners to hang in their schools

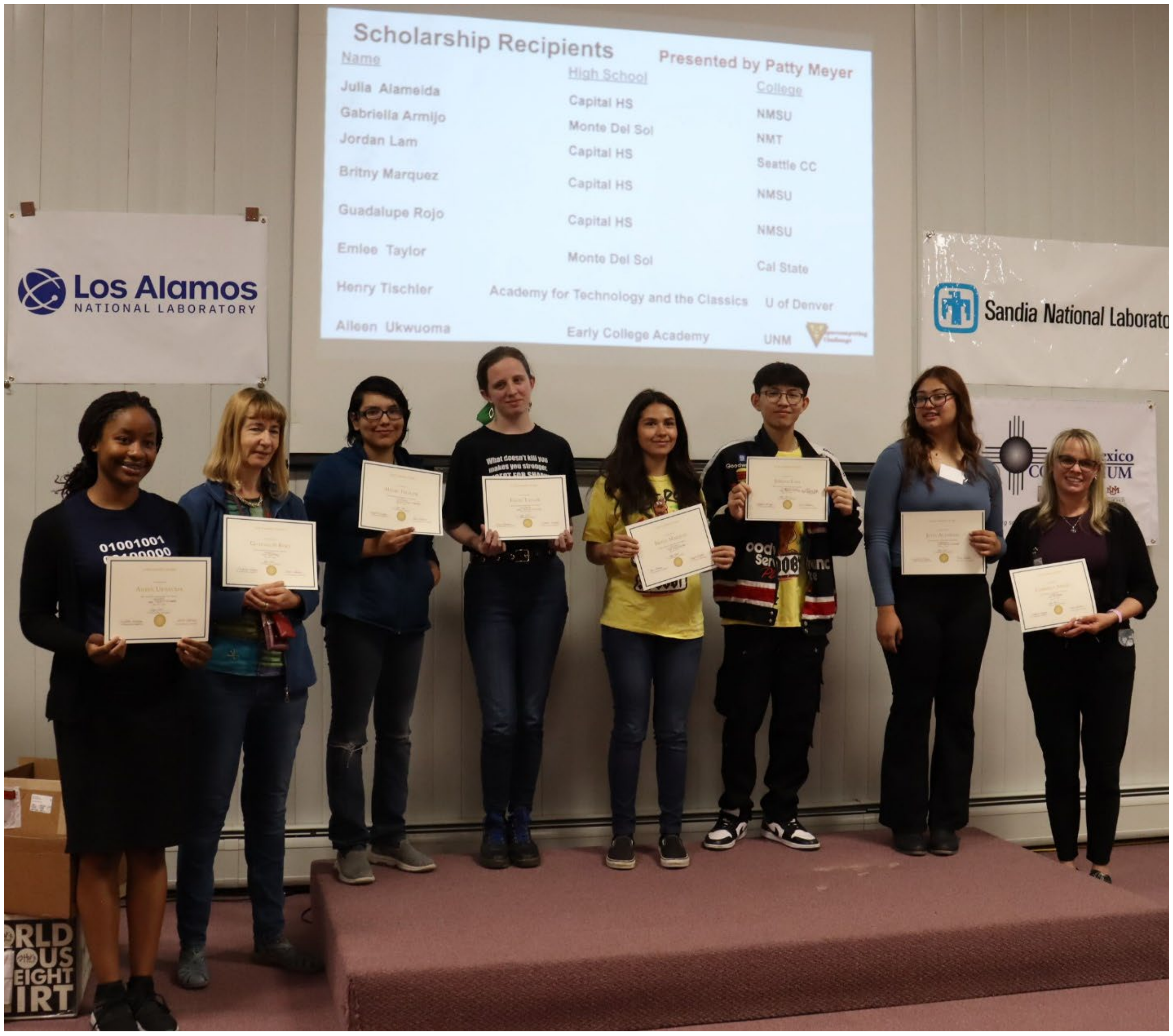
A complete list of all winning student teams.
https://supercomputingchallenge.org/23-24/expo-AllWinnersList.pdf

More about the Supercomputing Challenge

"The goal of the yearlong event is to teach student teams how to use powerful computers to analyze, model and solve real-world problems," said David Kratzer, Executive Director. "Participating students improve their understanding of technology by developing skills in scientific inquiry, modeling, computing, communications, and teamwork."

The New Mexico Supercomputing Challenge teaches written and oral communication, collaboration with peers and professionals, critical thinking including research and coding including computer modeling to middle and high school students throughout the state. Any New Mexico middle-school or high-school student, including home-schooled students are eligible to participate in the Supercomputing Challenge. Students follow their own interests to choose a topic to model.

Scholarship winners



Scholarship Recipients

Presented by Patty Meyer

Name	High School	College
Julia Alameida	Capital HS	NMSU
Gabriella Armijo	Monte Del Sol	NMT
Jordan Lam	Capital HS	Seattle CC
Britny Marquez	Capital HS	NMSU
Guadalupe Rojo	Capital HS	NMSU
Emlee Taylor	Monte Del Sol	Cal State
Henry Tischler	Academy for Technology and the Classics	U of Denver
Aileen Ukwuoma	Early College Academy	UNM

Scholarships worth $17,500 were awarded at the Supercomputing Challenge Awards Ceremony to eight seniors: Julia Alameida, Gabriella Armijo, Jordan Lam, Britny Marquez, Guadalupe Rojo, Emlee Taylor, Henry Tischler, and Aileen Ukwuoma.

All final reports are online.
https://supercomputingchallenge.org/23-24/final-reports-view

More information about the Supercomputing Challenge can be found at:
https://supercomputingchallenge.org

Teams Finishing the Challenge and submitting final reports:

School: The Academy for Technology and the Classics
Using Clustered Random Samples to Find Spurious Correlations in Neural Networks
Team Member: Henry Tischler
Sponsor: Jenifer Hooten

School: Albuquerque Academy, *Housing Problem*
Team Members: George Chekh, James Pulliam
Sponsors: Jay Garcia, Oksana Guba

School: Albuquerque Academy, *Water Runoff and Diversion Simulation*
Team Member: Harrison Schiek
Sponsors: Jay Garcia

School: Albuquerque Academy, *Single-Family Residential Thermal Energy Storage*
Team Member: Nicholas Allan
Sponsors: Jay Garcia

School: Capital High School
ROBOTIC GUNSHOT DETECTION SYSTEM: SCOOBY
Team Members: Raul Alvarado Villalobos, Craig Andree Abajo, Jesse Burch, Haize Christiansen, Jordan Lam, Britny Marquez, Edward Scott
Sponsor: Barbara Teterycz
Mentor: David Ritter

School: Capital High School, *Exploring the Science Behind High Explosives*
Team Members: Julia Almeida, Abigail Solorio, Guadalupe Rojo
Sponsor: Irina Cislaru
Mentor: Scott Elliott

School: V. Sue Cleveland High School, *MOCHA*
Team Members: Clark Kardian, Andrew Boggan
Sponsor: Ashli Knoell

School: V. Sue Cleveland High School, *A Realtime Camera Fusion 3D Model with a Novel Feature-Matching and Star Identification-Based Calibration for Tracking Smoke Plumes*
Team Member: Gene Huntley
Sponsors: Ashli Knoell, Leah Felty
Mentor: Stephen Guerin

School: V. Sue Cleveland High School, *Using Deep learning to save an endangered language*
Team Members: Layla Barela, Alexandra Sosa, Jack Arnaudville
Sponsor: Ashli Knoell

School: Harrison Middle School, *Farming the Unfarmable*
Team Members: Leland Martinez, Anthony Tapia, Anthony Amaya
Sponsor: Becky Campbell
Mentor: Michelle Fairow

School: Early College Academy/Justice Code
Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers
Team Member: Aileen Ukwuoma
Sponsor: Becky Campbell
Mentors: Robert Taylor, Justin Baca

School: Justice Code, *A Computational Exploration of COPD Medication Adherence*
Team Members: Samuel Campen, Alexandrina Ukwuoma, Aileen Ukwuoma,
Sponsor: Rebecca Campbell
Mentor: Daniel Fuka

School: Justice Code, *Flood Detectives: Using AI to Combat Global Warming*
Team Members: Mekhi Bradford, Delight Emma-Asonye, Leilani Baty
Sponsor: Becky Campbell
Mentor: Michelle Fairow

School: La Cueva High School, *CARE: Cancer And Radiation Education*
Team Members: Hadwyn Link, Ximena Serna, Iorwen Ouyang, Isaac Chen
Sponsor: Jeremy Jensen
Mentor: Mario Serna

School: La Cueva High School, *Smart Jacket*
Team Member: Paul Kotze
Sponsor: Jeremy Jensen

School: La Cueva High School, *AI Calling BS*
Team Member: Victoria Outkin
Sponsor: Jeremy Jensen

School: Los Alamos High School
Advancement in Marching Cubes for Enhanced Structural Design
Team Member: Andrew Morgan

School: Los Alamos High School, *Magnetic Reconnection*
Team Member: Tate D. Plohr
Mentor: Mark Petersen

School: Los Alamos High School
Facial Recognition with Deep Face Library - Ensuring Campus Security
Team Member: Hyunoo (Hayden) Kim
Mentor: Michela Ombelli

School: Los Alamos Middle School, *DUONUMERO: MULTILINGUALISM IN MATH*
Team Member: Linus Plohr

School: McKinley Middle School, *Mushrooms as Reducers of Trash*
Team Member: Lennon Martinez, Elias Villa, Alexandrea Braden
Sponsor: Sharee Lunsford
Mentors: Karen Glennon, Patty Meyer

School: Monte del Sol Charter School, *SCRAM*
Team Members: Gabriella Armijo, Adelina Baca, Emlee Taylor-Bowlin
Sponsor: Rhonda Crespo

School: New Mexico Academy for the Media Arts
The Wood-Wide Web's Impact on Plant Health in Arid Areas
Team Members: Eduardo Dorado, Ana Sofia Rodriguez, Elijah Nasser-Sparks, Wyatt Wade
Sponsor: Tanya Mueller

School: New Mexico School for the Arts, *The Impact of Microplastics on Algae (Respiration)*
Team Members: Gloria Galassi, Elisea Jackson
Sponsor: Sarah Rowe

School: Santa Fe Preparatory School
Computational Hydrodynamic Analysis for Speed Maximization (CHASM)
Team Members: Luke Rand, Greta Swanson, Nandita Ganesan
Sponsor: Jocelyn Comstock

School: Santa Fe Preparatory School
A Systems Approach to Understanding Artificial Night at Light
Team Member: Camila Carreon
Sponsor: Jocelyn Comstock, Gabriella Masoni
Mentor: Mark Galassi

School: St. Thomas Aquinas School, *Spacecraft Debris Avoidance in a 3D Environment*
Team Members: Catherine Sedillo, Amelia Sedillo
Sponsor: Eric Vigil
Mentor: James Sedillo

School: St. Thomas Aquinas School
The Impact of Sharks Disappearing in the Great Barrier Reef
Team Member: Victoria Reichow
Sponsor: Eric Vigil

School: Truman Middle School
Robots on a Mission: Safeguarding Animals from Wildfire Disasters
Team Members: Vincent Nuanes, Josue Ochoa, Ayla Alaya
Sponsor: Natali Barreto Baca

School: Truman Middle School, *The Sky's Watchmen: Using Drones to Safeguard Forests*
Team Members: David Chavira, Carlos Cantu
Sponsor: Natali Barreto Baca

School: Tucumcari High School, *Color Detection*
Team Members: Sariah Mardo, Rachel Mardo
Sponsor: Thomas Evans

School: Tucumcari High School, *Effects of Weight on Aircraft and Aviation*
Team Members: Nolan Ryen, Marcus Lopez
Sponsor: Thomas Evans

School: Welch Homeschool, *Virtually Reconstructing an Ancient Musical Instrument*
Team Member: Helena Welch
Sponsor: Cindy Welch
Mentor: Paul Welch

School: Welch Homeschool, *Detecting Exoplanets Through Transits*
Team Member: Kalliope Luna Welch
Sponsor: Cindy Welch
Mentor: Paul Welch

Judges

Ed Angel, Professor Emeritus UNM
Char Arias, Sandia National Laboratories/Retired
Gennie Barrett, Talking Talons Youth Leadership
Richard Barrett, Sandia National Laboratory/Retired
Nicole Batey, CURIA Global
Nick Bennett, TLG Learning
Thomas Bowles, Los Alamos National Laboratory
Vincente Bravo-Valencia, Westwind Computer Products Inc.
Hope Cahill, Santa Fe Public Schools
Noah Caulfield, Supercomputing Challenge alumni
Yie Sheng Chen, University of New Mexico
Matthew Curry, Sandia National Laboratories
Alan Ray Daugherty, Melrose Municipal Schools

Mike Davis, HPE/Cray Inc.
Rusty Davis, Los Alamos National Laboratory
Sharon Deland, Sandia National Laboratories/Retired
Nelson Dsouza, Microsoft
Mohit Dubey, Los Alamos National Laboratory, Supercomputing Challenge alumni
Creighton Edington, Rural Education Advancement Program
Drew Einhorn
Michelle Fairow, University of New Mexico
Meg Fisher, Apple/Retired
Susan Gibbs, Project GUTS
Stephen Guerin, Simtable
Essa Imhmed, Eastern New Mexico University
Omar Ishak, Los Alamos National Laboratory
David Janecky, Los Alamos National Laboratory/Retired
Ian Jensen, Microsoft
Elizabeth Jimenez, University of New Mexico
Philip Jones, Los Alamos National Laboratory
Carene Larmat, Los Alamos National Laboratory
Brandon Lattimore, Los Alamos National Laboratory
Maximo Lazo, University of New Mexico
Scott Levey, Sandia National Laboratories
Monique Morin, Sandia National Laboratories
Joseph Olonia, Central New Mexico Community College
James Overfelt, Sandia National Laboratories
Mark Petersen, Los Alamos National Laboratory
Nick Porter, University of New Mexico
Paige Prescott, Computer Science Alliance
Maureen Psaila-Dombrowski, Los Alamos National Laboratory
Naomi Rankin, Johns Hopkins University, Supercomputing Challenge alumni
Lonnie Rednour, San Juan College
Dana Roberson, Central New Mexico College
Teri Roberts, Los Alamos National Laboratory/Retired
Thomas Robey, Gaia Environmental Sciences
Mary Sagartz
Ryan Scherbarth, University of New Mexico
Sarbagya Shakya, Eastern New Mexico University
Reffat Sharmeen, Sandia National Laboratory
Tim Thomas, Sandia National Laboratories
Michael Trahan, Sandia National Laboratories
Geoff Valdez, Los Alamos National Laboratory
Eduardo Ceh Varela, Eastern New Mexico University
Blaine Vigil, Los Alamos National Laboratory
Anneliese Ward, University of New Mexico
Rachel Washington, University of Colorado Boulder, Supercomputing Challenge alumni

Do you want to become a supporter of the Supercomputing Challenge?
Please email us at consult@supercomputingchallenge.org for details.

Magnetic Reconnection

School:
Los Alamos High School
1300 Diamond Dr.
Los Alamos, NM 87544

Team Members:
Tate D. Plohr

Project Mentor:
Dr. Mark Petersen

MAGNETIC RECONNECTION

Tate D. Plohr

ABSTRACT
Magnetic reconnection is the reconfiguration of magnetic field lines in plasma. It is a phenomenon that drives solar flares and Coronal Mass Ejections (CMEs) on the Sun's surface. Strong CMEs and solar flares are capable of affecting Earth by damaging electrical equipment, disrupting communication, and harming astronauts in space. In my project, I wrote a Python program that simulates the onset of magnetic reconnection. I have used my program to understand conditions that lead to magnetic reconnection. In summary, I have found that increased magnetic field strength and increased plasma resistivity stabilize a current sheet and consequently facilitate magnetic reconnection. This understanding is essential for developing predictive models of magnetic reconnection that help mitigate the catastrophic damages caused by CMEs or solar flares.

tdplohr@gmail.com, Los Alamos High School, Los Alamos, NM

INTRODUCTION

On the surface of the Sun, many powerful phenomena occur. As fascinating as they are, they have the potential to harm people on Earth. Massive arches of plasma, called coronal arches, constantly generate solar wind that bombards the Earth's magnetic field. In addition, if they are big enough, the arches can create solar flares and Coronal Mass Ejections (CMEs), which are even more powerful. Solar flares and solar wind emit strong electromagnetic radiation, including radio waves and ultraviolet radiation, which can disrupt communication and harm astronauts in space as well as people at high altitudes in airplanes. Along with radiation, plasma from the sun's surface can be propelled towards Earth by CMEs. If the plasma hits Earth, it can cause electrical failures and cut communication lines. Such an event could be disastrous in many ways. For instance, in hospitals without secure or backup power, life-supporting devices could become dysfunctional. For the United States alone, it is estimated that a CME could cost tens of billions of dollars [1]. For example, in September 1859, a massive CME hit Earth and caused the Carrington Event. The Carrington Event was so strong that auroras, which are a result of the interaction between Earth's magnetic field and solar plasmas, were seen near the equator, compasses failed to point North, and telegraphs threw sparks and transmitted messages without a battery connected [2]. A CME on the scale of the Carrington event or larger would undoubtedly cause more serious complications in our modern world, where much more than telegraph lines are powered by electricity.

Therefore, it is very important to mitigate the damage caused by these phenomena. Doing so requires early prediction of a large solar flare or CME [3]. However, our prediction methods are currently only able to detect solar flares or CMEs right before they start coming towards us. We have little time to prepare.

The mechanism behind these dangerous phenomena that drives the massive burst of energy is magnetic reconnection [4] (See Fig. 1.). Magnetic reconnection occurs when two opposing magnetic field lines reconfigure to form a new magnetic system. This releases tremendous energy by converting magnetic potential energy into kinetic energy of the coronal plasma [5]. The accelerated plasma creates solar wind, solar flares, and CMEs. Understanding the dynamics of magnetic reconnection, particularly its onset, is vital to predicting when a solar flare or CME will form.

The standard model of magnetic reconnection is the Sweet-Parker model, also known as a current sheet or magnetic separatrix. Figure 1 is a schematic diagram of the process. Each of the three stages also represents the three main topics of magnetic reconnection; (a) whether or how two magnetic fields come close to initiate reconnection, *i.e.*, onset problem; (b) how fast reconnection happens, *i.e.* rate problem; (c) how much magnetic energy will be converted to kinetic energy, *i.e.* energy partition problem.

The standard model of Magnetic Reconnection is called the **Sweet-Parker model**, also known as a **current sheet**.

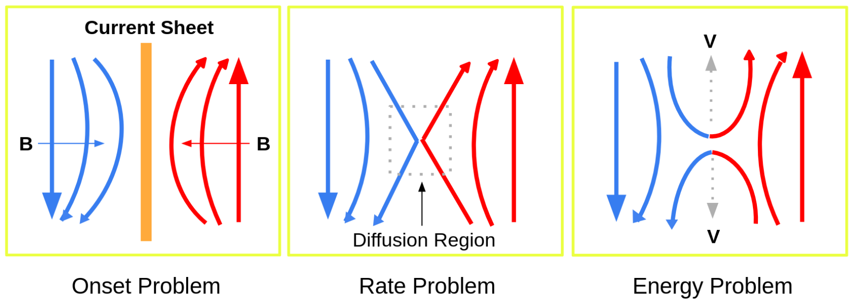


Figure 1. Schematic diagram of magnetic reconnection
(a) Two magnetic fields come toward each other; (b) via diffusion they reconnect;
(c) the new field lines carry away plasma (with very high energy).

In this paper, we are interested in the stability of a current sheet to see its validity as a model for the onset of magnetic reconnection. Hence our focus is on (a) in Figure 1. We assume that the flow is incompressible, i.e., the mass density is uniform, because it has been shown that compressible flow is more stable [6]. An incompressible current sheet is described by two variables: the fluid velocity and the magnetic field. In an idealized configuration, both are in the vertical direction.

The velocity field of the current sheet is called the Bickley Jet in fluid dynamics (when there is no magnetic field) and is known to be unstable. There is a name for the kind of instability that the jet is subject to: the Kelvin-Helmholtz instability. With a sinusoidal perturbation in the vorticity field, chains of vortices spread out from the jet to the left and right, as shown in the figure below.

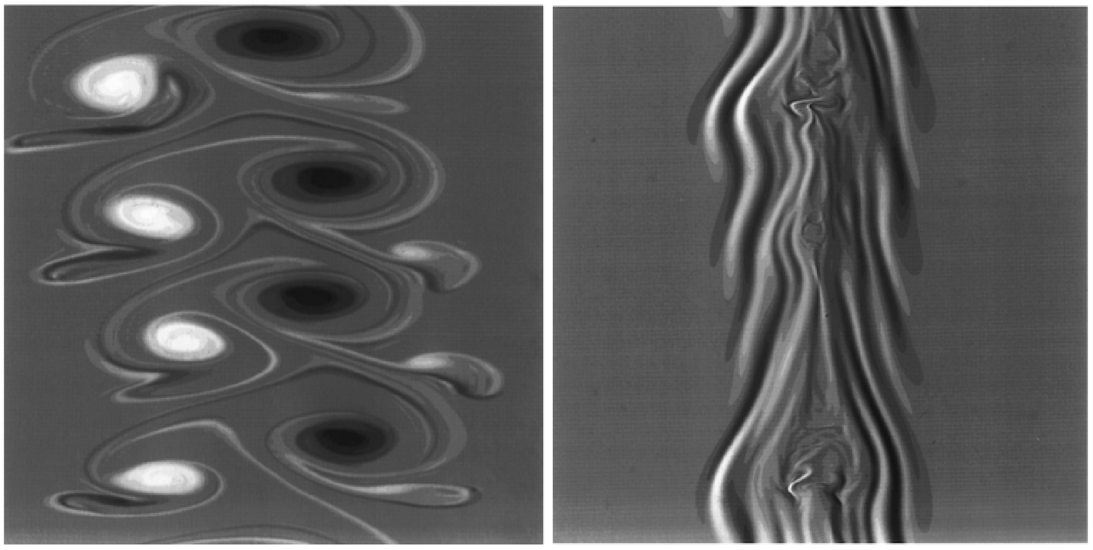


Figure 2. (a) Bickley Jet evolution of sinusoidal perturbation without magnetic field. [6]
(b) Current Sheet evolution of sinusoidal perturbation with magnetic field. [6]

Since the current sheet has the Bickley Jet as the velocity profile, one might expect that the current sheet would also be unstable. However, the surrounding magnetic field acts as a guide for the jet and stabilizes the flow. This result was shown in a paper by Biskamp et al. [6].
The goal of this paper is to perform a systematic study on the evolution of the current sheet. I aim to find how a current sheet is affected by a perturbation in the velocity field and the effects of the resistivity and the magnetic field.

METHODS

I wrote a computer program in Python to simulate a current sheet with a perturbation. My code uses two differential equations to describe the system and compute the next time step. To solve the equations, I used the centered finite difference method to calculate derivatives. The simulation uses mixed boundary conditions: periodic for fluid flow and Neumann for the magnetic field. The aspect ratio of the system was 2:1; there were 128 grid points in the x direction and 64 in the y direction.

The equations governing the flow are written in terms of the vorticity ω (omega) and the magnetic flux function ψ (psi). Because the flow is incompressible, the fluid velocity is perpendicular to the gradient of the velocity potential ϕ (phi), which is obtained from the vorticity by solving the Poisson equation shown below. Likewise, the magnetic field is perpendicular to the gradient of the magnetic flux function ψ (psi). The governing equations are as follows [6]:

$$\partial_t\omega + \mathrm{v} \cdot \nabla\omega = \mathrm{B} \cdot \nabla j + \mu\nabla^2\omega, \qquad \text{Eq. (1)}$$

$$\partial_t\psi + \mathrm{v} \cdot \nabla\psi = \eta\nabla^2\psi, \qquad \text{Eq. (2)}$$

where the velocity, magnetic field, vorticity, and current density are defined by

$$v = e_z \times \nabla\phi \qquad B = e_z \times \nabla\psi$$

$$\omega = \nabla^2\phi \qquad j = \nabla^2\psi$$

In these equations, μ (mu) is fluid viscosity and η (eta) is magnetic resistivity.

Computer codes need verification tests to make sure they are written correctly and produce sensible results. For this purpose, I wrote several versions of the code, adding capabilities for each update.

To speed up the code, I used GPU parallelization to compute the solution on GPU processors. I used the Python library CuPy to speed up the code by converting my Numpy arrays into CuPy arrays that are processed by the GPU.

In the following subsections, I will explain the verification and optimization methods I have used.

Verification

In Eqs. (1) and (2), the fluid motion is described by diffusion and advection. The diffusion term is the second derivative in space and advection is the first derivative in space, multiplied by velocity. I tested the diffusion and advection terms separately in 1D and 2D settings. Then I tested the full equations, having both diffusion and advection activated.

(1) In a very simple version of my code, I had a one-dimensional system described by Eq. (1) with only the diffusion term. The initial condition was a sine wave. The system has an analytical solution:

$$u_{ext} = A \sin{(2\pi x/L)} e^{-4\mu\pi^2 t/L^2}$$

Running my code, I expected to find an answer equal to the exact solution. There was some discrepancy due to numerical approximations in the calculations. I found that the root mean squared (RMS) difference between the numerical and exact solutions was of order 0.1%.

(2) I solved Eq. (1), again with just the diffusion term, in two dimensions. The initial conditions were, however, quasi-1D, meaning there was no variation in the y direction. For each value of y, the RMS difference between the numerical and exact solutions was of order 0.1%.

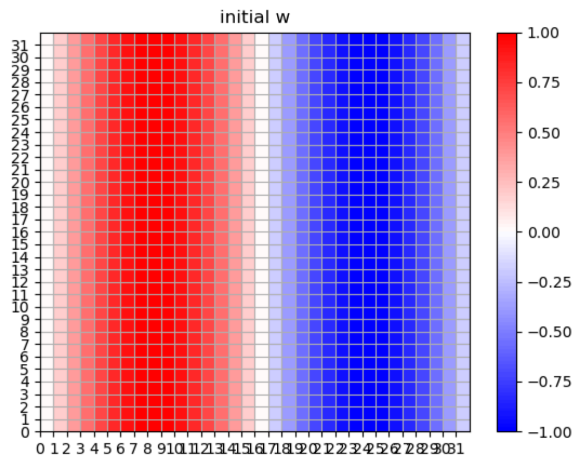


Figure 3. Initial vorticity profile for (2). X and y are spatial dimensions.

(3) The previous tests only involved the diffusion term. The next test used only the advection term. Equation (1) is used in two dimensions without the diffusion or magnetic term. In this test, to find the velocity, my code solves the Poisson equation for ϕ. The initial condition was a circular Gaussian bump in the vorticity, ω. Because the vorticity field was circular, the velocity is angular and the advection term should be zero. The RMS difference between the vorticity at the final and initial times was of order 0.0001%.

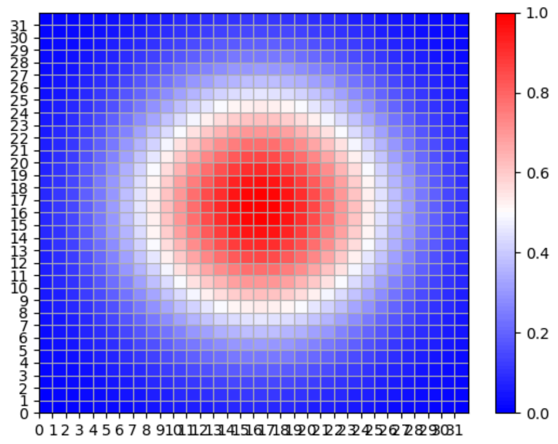


Figure 4. Initial vorticity profile for (3). X and y are spatial dimensions.

(4) I again tested the advection term in Eq. (1), but now with an added diagonal velocity. Because of the periodic boundary conditions, the bump should move diagonally, exit through the top right corner, enter through the bottom left corner, and return to the center. The RMS difference was again calculated between the final time step and the initial condition. The difference was of order 1%, which is relatively large. It could be reduced by using a better time-stepping method or by refining the grid. Nonetheless, the difference is small enough to show that my code is correctly solving the equations and the difference is due to numerical approximations.

(5) Next, I tested the advection term without an added velocity, meaning that the velocity field is induced by the vorticity. I tested my code with two vortices. The flow is too complex to have an analytic solution, so the simulation was verified only by observing its qualitative behavior. These initial conditions are well-known in fluid dynamics, so I knew that the two vortices should merge at the center after swirling around each other for a while.

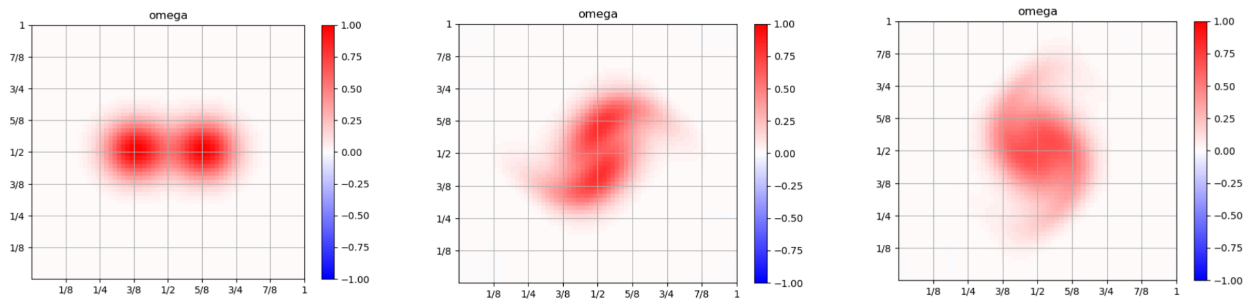


Figure 5. Plots of vorticity showing the evolution through time. X and y are spatial dimensions.

(6) I tested my code using the initial conditions of the Bickley Jet. This is the velocity profile of the current sheet, which is known to be a steady-state but unstable solution. Steady-state means that the system should not change in time. The RMS difference was calculated between the final time step and the initial condition. The error was of order 0.1%.

Procedure

The velocity and magnetic field profiles in a current sheet are modeled by:

$$v_y(x,y) = V_0 \text{sech}^2(x),$$

$$B_y(x) = B_0 \tanh(x),$$

from which I calculate vorticity and the flux function as

$$\omega(x,y) = V_0 \frac{\tanh(x)}{\cosh(x)},$$

$$\psi(x) = B_0 \ln(\cosh(x)).$$

Then I apply a perturbation to the vorticity field to get the initial conditions for the simulations:

$$\omega(x,y) = V_0 \frac{\tanh(x)}{\cosh(x)} * pert$$

with $pert = \text{A} \sin(ky)$

I set the code to run 1,800 time steps and to plot intermittently.

Optimizations

My code, like other Direct Numerical Simulations (DNS), takes significant computational power and time. Since several runs are needed to get the results I present, optimization of my program is necessary. It drastically lowers the effort needed and quickens debugging, which takes the majority of development time. In my project, several strategies were followed to reduce the run time of my program. In the first full version of my code, the run time for a full simulation took almost 2 hours. Using the NumPy library, I vectorized my equations. When, for example, adding two arrays, vectorization makes the computer add every entry at once rather than one at a time. For my program, this reduced the run time to 1 ½ hours. Then, with a library called CuPy, I utilized the GPU to parallelize my program, which reduced the runtime to 1 hour. GPU parallelization involves spreading calculations among many GPU processors. Normally, the computer uses the CPU, which has only a few processors, each very powerful. In contrast, GPU processors are relatively weak, but there are thousands of them. For a DNS, the GPU is much more efficient. Finally, by using a sparse matrix solver in the SciPy library, I reduced the computational time to 8 minutes. To solve the Poisson equation that describes the relationship between ϕ (phi) and ω (omega), $\omega = \nabla^2\phi$, I represented the Laplacian operator (∇^2) as a sparse matrix and used a sparse matrix solver. After this conversion, the runtime was drastically reduced.

RESULTS

The equations in my code use vorticity to describe the flow of the plasma instead of velocity; this approach simplifies the calculations. Vorticity is a measure of swirling motion, like in a hurricane. A hurricane is a vortex in which the wind swirls around the center in a circular motion. A positive value of vorticity indicates a vortex that is spinning counter-clockwise, whereas a negative value corresponds to a clockwise spin.

The first series of plots (Figure 6) shows the vorticity profile of the perturbed current sheet without a magnetic field. The second series of plots (Figure 7) has a magnetic field with an amplitude of 0.3, which leads to different results for the same initial vorticity profile.

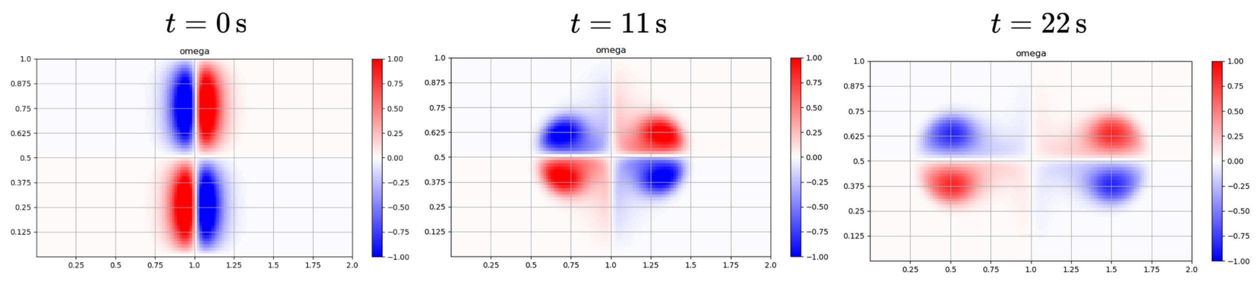


Figure 6. Evolution of vorticity in the non-magnetized current sheet. X and y are spatial dimensions

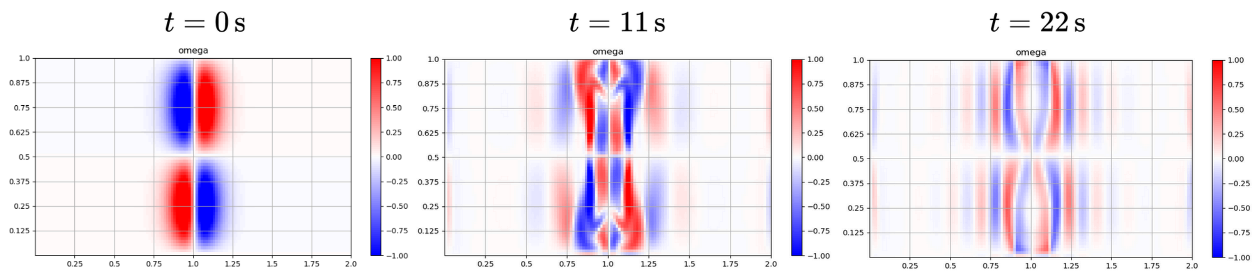


Figure 7. Evolution of vorticity in the magnetized current sheet. X and y are spatial dimensions

Furthermore, I ran a parameter study of the current sheet by varying the resistivity and the magnetic field strength (Figure 8). Each plot in Figure 8 shows the vorticity profile of the current sheet after 22.5 seconds.

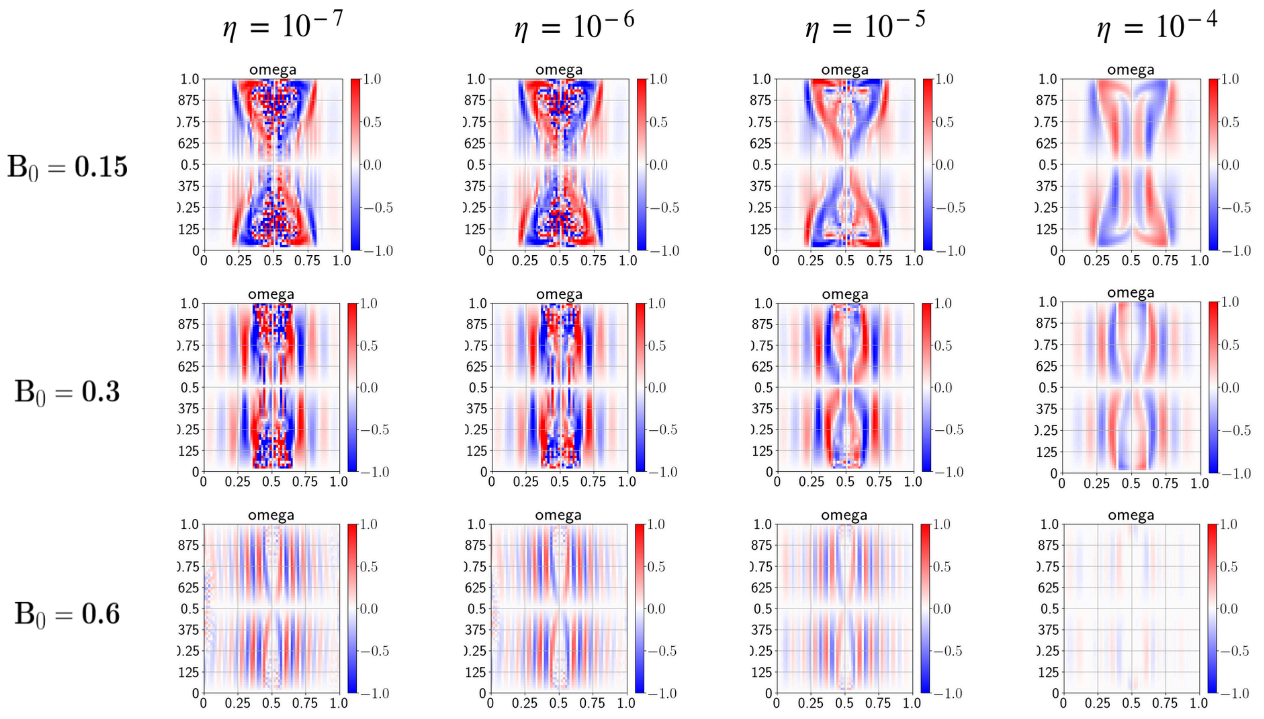


Figure 8. The final vorticity profile with different values of resistivity and magnetic field strength. X and y are spatial dimensions.

I then calculated the L2 norm of the vorticity profiles to make a quantitative measure of stability:

$$\|\omega\|_2 = \sqrt{\frac{\sum_{i,j}\left(\omega_{i,j}\right)^2}{N}},$$

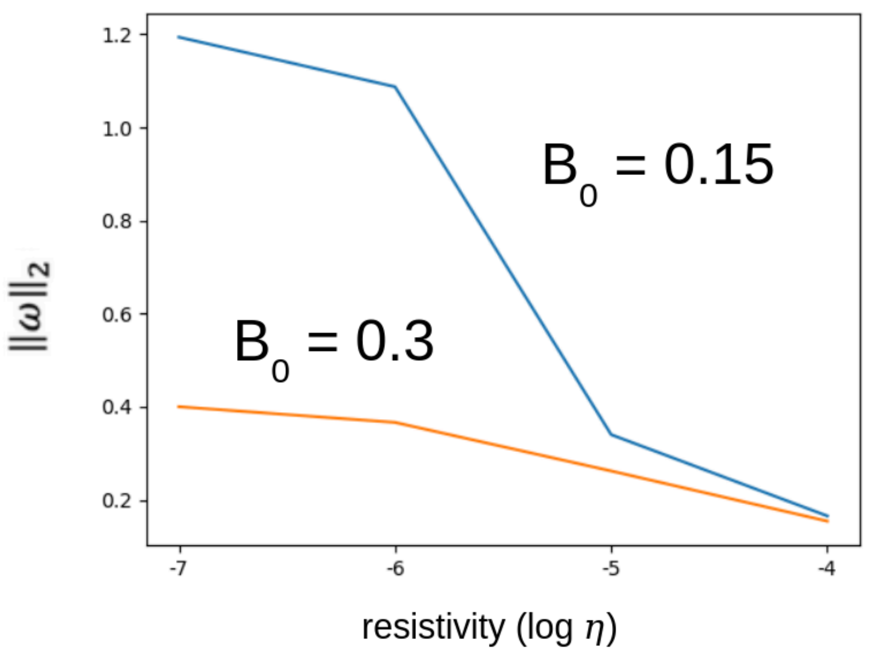


Figure 9. L2 norm of the vorticity profiles. The blue line is for $B_0 = 0.15$ and the orange line is for $B_0 = 0.3$
For a fixed magnetic field strength, an increase in resistivity leads to a lower L2 norm of vorticity. This implies that the flow is more stable.

Here I show the average magnetic energy at the final timestep for different initial conditions:

$$e_B = \frac{B^2}{2\mu_0}$$

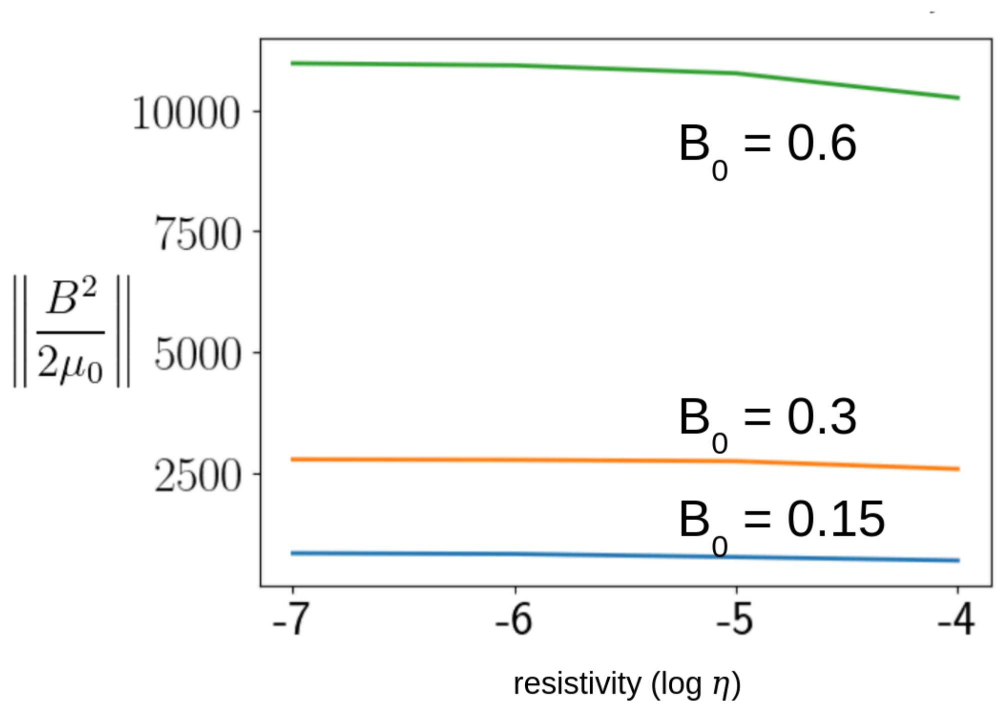


Figure 10. Average magnetic energy. The blue line is for $B_0 = 0.15$, the orange line is for $B_0 = 0.3$, and the green line is for $B_0 = 0.6$.

CONCLUSIONS

In the first series of plots (Figure 6), pairs of vortices push out to the sides due to the perturbation; this shows that the system is unstable to the perturbation. The cause is that the pairs of vortices induce advection. Like a hand mixer, a pair of opposite sign vortices push fluid behind them, so that they are pushed forward. This leads the two pairs of vortices to collide in the center and then spread outwards.

In the second series (Figure 7), the results are strikingly different. The jet merely broadens, showing the same qualitative behavior as in the paper by Biskamp et al. [6]. The magnetic field stabilizes the flow.

After confirming that the magnetic field stabilizes the current sheet, I further tested the current sheet by varying the strength of the magnetic field and the plasma resistivity. As mentioned earlier, each plot in Figure 8 is the vorticity profile of the current sheet at 22.5 seconds. As can be seen in the plots, increasing the resistivity (η) has a stabilizing effect. As well as the resistivity, the magnetic field strength also has a huge effect. Also, note that the difference between the profiles for $B_0 = 0.6$ is very small despite the changes in resistivity.

This is further demonstrated by the L2 norm graph (Figure 9). As the resistivity is increased, the L2 Norm goes down, meaning a less turbulent vorticity profile. The orange line, for $B_0=0.3$, has less turbulence than the blue line, for $B_0=0.15$, again showing that the magnetic field stabilizes the current sheet.

From Figure (10) we find that magnetic energy is mainly determined by the initial magnetic field strength and is relatively insensitive to the resistivity.

These results show that current sheets with strong enough magnetic fields and high enough resistivity are stable to perturbation while maintaining high magnetic energy. In the case of coronal arches, both are strong enough to make the current sheet stable. If the current sheets were unstable, magnetic reconnection would not occur according to the Sweet-Parker model. The stability of the current sheet facilitates magnetic reconnection to occur. My project shows that the Sweet-Parker model can describe magnetic reconnection for when coronal arches form solar flares and Coronal Mass Ejections.

BIBLIOGRAPHY

[1] Solar storms could cost USA tens of billions of dollars. University of Cambridge. (2017, January 19). https://www.cam.ac.uk/research/news/solar-storms-could-cost-usa-tens-of-billions-of-dollars

[2] May, A., & Dobrijevic, D. (2022, May 20). *The carrington event: History's greatest solar storm*. Space.com. https://www.space.com/the-carrington-event

[3] Laboratory, L. A. N. (n.d.). Forecasting space weather: Discover los alamos national laboratory. LANL Discover RSS. https://discover.lanl.gov/publications/national-security-science/2022-spring/vania-jordanova/

[4] Coronal mass ejections. Coronal Mass Ejections | NOAA / NWS Space Weather Prediction Center. (n.d.). https://www.swpc.noaa.gov/phenomena/coronal-mass-ejections

[5] Gurnett, D. A., & Bhattacharjee, A. (2017). *Introduction to plasma physics: With space, laboratory and Astrophysical Applications*. Cambridge University Press.

[6] Biskamp, D., Schwarz, E., & Zeiler, A. (1998). Instability of a magnetized plasma jet. Physics of Plasmas, 5(7), 2485–2488. https://doi.org/10.1063/1.872931

ACKNOWLEDGEMENTS

I want to thank my mentors for their support in my project and in writing this paper.

Mark Petersen guided me through writing my code and provided helpful insights.
JeeYeon Plohr helped with the research and execution of my project as well as the presentation.
Bradley Plohr helped me conceptualize much of the mathematics required.

Albuquerque Academy Team 1

Water Runoff and Diversion Simulation

New Mexico

Supercomputing Challenge

Final Report

March 22, 2024

AA Team 1

Albuquerque Academy

Created by

Harrison Schiek

Teacher Mentor

Jay Garcia

Table of Contents --------------------------

Executive Summary ----------------------

Problem

As Global Warming becomes increasingly more prominent, the weather in Albuquerque has gotten more and more extreme. This year, random bursts of snow and precipitation marked another unusually warm spring. For communities all over the city, and especially those close to the river, flash flooding and weather related property damage are increasingly more of a threat. As the climate continues to change, our water diversion and urban/suburban infrastructure must rise to meet the challenge. But how will we know what needs attention for the next downpour?

Solution and Methodology

My simulation aims to help by predicting what parts of our city and water diversion network need work to truly protect the residents of Albuquerque. The simulation calculates flow and accumulation and marks areas where excessive water collects or where the water flows too fast as problem areas that need some form of intervention to be safe. It also measures the total water absorbed to determine the effects of various changes on the recharge of the aquifer. My model reconstructs the topography through a grid elevation approximation. It then iterates through each grid cell, applying the given rainfall and absorption and moving the water according to elevation and current water depth.

Validation and Verification

With basic geometries as a foundation, my model reconstructs flow based on equations of incline flow over time. Data from The Albuquerque Journal for a flash flood in the Embudo Arroyo (2021) will be used as a real scenario to test against.

Results

My model predicts important areas of Albuquerque that would benefit from redesign and improvement of runoff management infrastructure. The Snow Heights neighborhood, as well as a large region of riverside communities aren’t sufficiently prepared for storms that would put pressure on their current systems. Another significant issue is erosion in both the mountains and in underdeveloped areas where the soil has less vegetation to secure it.

Problem ------------------------------------

All over the world, temperatures have risen over the past century. As a result, more and more water is being evaporated from the oceans, dumping huge amounts of water into local water cycles[1]. Increasing numbers of hurricanes are hitting both of the coasts of the US, and with this, more and more storms are continuing inland towards the dry and arid landscapes of New Mexico and its neighboring states. These storms threaten to overrun the rustic water infrastructure that already seems insufficient. There were 1210 flash floods in the years 1993 to 2017 in New Mexico[3]. It is essential that communities all across the traditionally arid west work to overcome these sporadic, unpredictable, and extreme storms when they do come, effectively preventing both erosion and harmful accumulation. Flash floods are the most costly severe weather in the entire US, causing $5 billion in damages a year[11]. Additionally, these increasingly common extreme weather events have begun to pose a significant threat to the US agricultural industry[2]. The management of erosion and flood waters is crucial to maintaining the local New Mexican agricultural industry as well.

On top of that, Albuquerque's aquifer is always in need of additional supply for the hundreds of thousands of New Mexicans who rely on it. Albuquerque needs to ensure that these new storms can most effectively replenish our dwindling supply. Although aquifer levels are making a bounce back due to effective conservation efforts[4], truly securing these water resources for all residents of Albuquerque requires considering additional means to revitalize the aquifer. By reducing reliance on the Rio Grande through accelerating the shift to complete aquifer sustainability, we will allow more of those water resources to go to people in need.

Methodology and Algorithms -----------

I developed my water runoff model in python in Anaconda/Jupyter Cloud Notebooks for the majority of the project, switching to BBEdit and my computer's terminal in the last stretch. I built my model completely from scratch. To produce many of the final results, I ran my simulations on a desktop computer (Alienware AMD Ryzen 7 5800 3.4 GHz 8 Core Processor) which was significantly more powerful than the Macbook Air (1.1 GHz Dual-Core Intel Core i3)

I used for much of the initial development. The two computers both able to run my model significantly helped in getting all the verification, validation, and results simulations.

My simulation uses a square grid approximation for the desired topography (Albuquerque in this case). The elevation is input from a .csv file of restructured data from USGS ⅓ Arc second DEMs[12] (Figure 1 shows this elevation data; however, the simulations later on run on a subset of this data (boxed in red), as the full set proved too slow for the number of simulations needed). My program adds water over the area to simulate rainfall. A portion is subtracted to simulate absorption. Soil can absorb water fastest when it first encounters water, but then slows as the topsoil becomes saturated and the only absorption is from space made by water draining down from those saturated layers to lower layers. My simulation approximates that process according to absorption (A) as a function of time in minutes (t) with initial absorbance a_i:

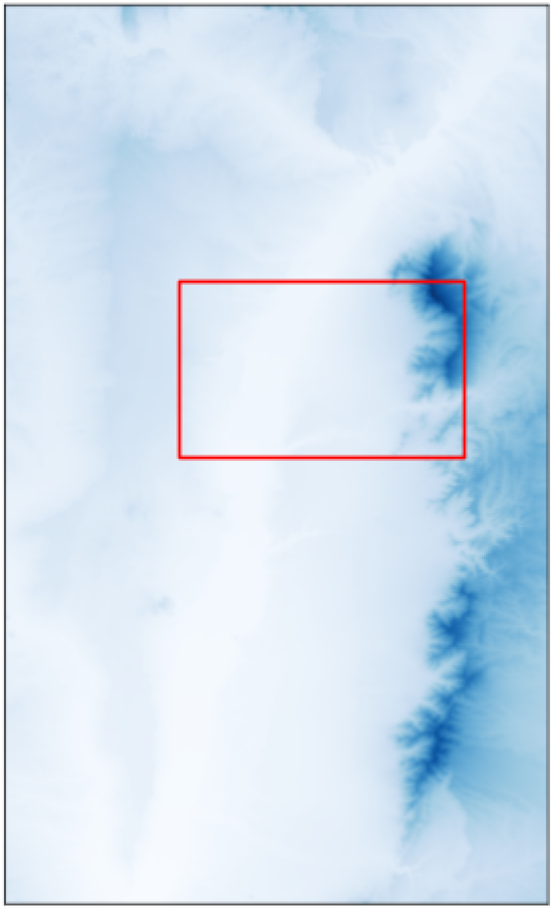
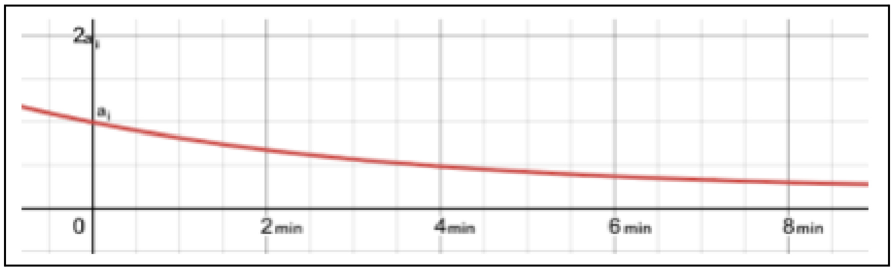


Figure 1 - Scale: 1437.7 - 3254.7 m

$$A(t) = \frac{4}{5}a_i e^{-\frac{t}{4}} + \frac{a_i}{5}$$

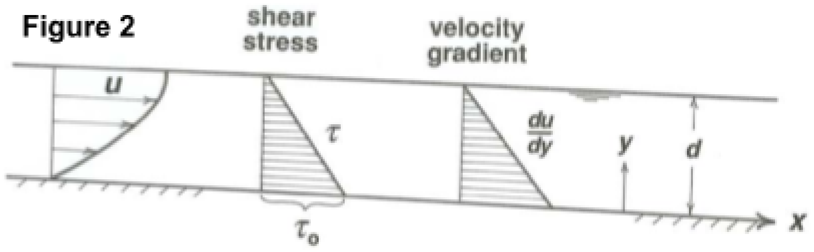


As time goes on, the absorption rate goes to ⅕ of a_i as seen in the accompanying graph. It is important to note that this is an assumption and not verified. This is due to the difficulty of making generalizations about a sample size as large as the Albuquerque area where soil types and compositions can vary drastically from one subset to another. However, it is accurate in that the absorbance of some soil starts relatively high then trends down to some value called (for the whole system) the steady state drainage rate. My model gets a_i from

USDA data on Hydrologic Soil Group[8] for Albuquerque and the surrounding Areas. However, due to issues normalizing the two data sets, I made a generalization of Hydrologic Soil Group to elevation (which are quite strongly correlated for much of the Albuquerque Area). The Hydrologic Soil Group can then be converted to an infiltration rate[9]. This allows my model to directly apply absorbance data based on the preexisting elevation data without the need for normalization.

Once one step of rain and absorption is complete, my program calculates the individual flow between grid squares. To speed up the algorithm, the portions of flow are calculated before the main program begins (a process from here on referred to as precalculation). This is to prevent the algorithm from calculating the way the flow is sent 100 times when it could have only been calculated once. The flow itself is given by a formula from Newtonian fluid dynamics[5]:

Figure 2

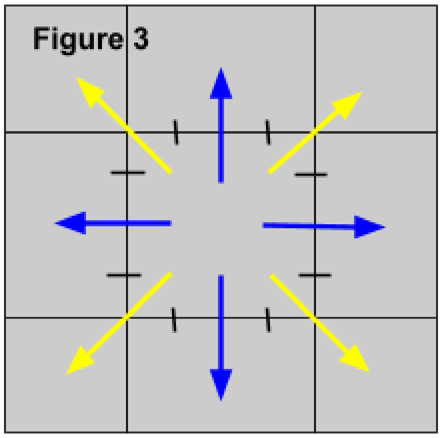


$$u = \frac{\gamma \sin\alpha}{\mu}\left(yd + \frac{y^2}{2}\right)$$

Where u is the flow rate at a given point (in m/s), γ is the gravitational force per unit volume of the substance (for water: $9.8\text{m/s}^2 * 1\text{g/cm}^3$ or $9800000\text{g/m}^2\text{s}^2$), μ is the dynamic viscosity ($1.308 mPa \cdot s$ or 1.308g/ms assuming the water is at 10ºC (or 50ºF))[6], d is the depth of water (cm for my model), and y is the variable for how deep in the depth of water you are (For all letters see Figure 2) . By integrating the expression with respect to y from y=0 to y=d (the whole depth), we get an expression for the flow rate per width:

$$\int_0^d u\, dy = \frac{\gamma \sin\alpha}{\mu}\int_0^d \left(yd + \frac{y^2}{2}\right)dy = \frac{\gamma \sin\alpha}{\mu}\left(\frac{d^3}{2} + \frac{d^3}{6}\right) = \frac{2\gamma \sin\alpha d^3}{3\mu}$$

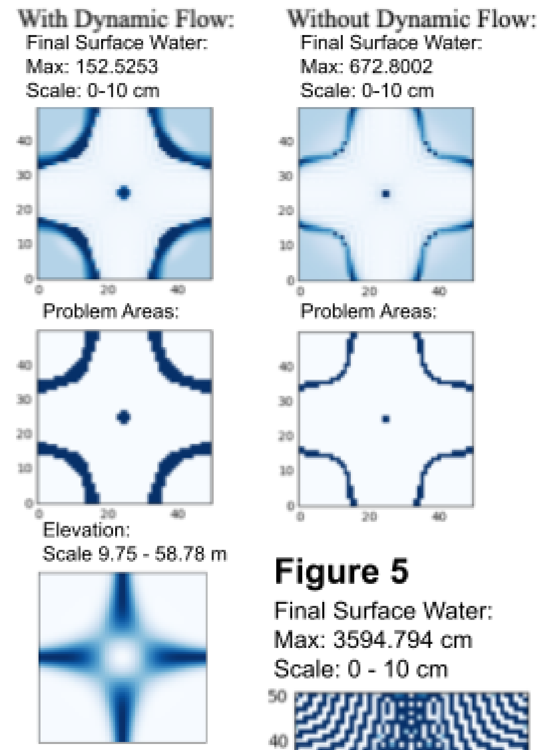
However, since it is giving only the flow per width (hence the strange units of m^2/s), there is one last manipulation. That last step is multiplying by the length of the flowing side which, since a tile has 8 ways to flow and a perimeter of $4 \cdot tileDim$ (side length will be referred to as tileDim), is ½ tileDim (As seen in Figure 3: each path of flow receives half of a side length to flow through). So the total flow for one direction is $\frac{\gamma \sin\alpha d^3 \cdot tileDim}{6\,\mu}$. Finally, since my model uses



depth rather than volumes, the total volumetric flow needs to be divided by the area it affects to get a change in depth: flow for one direction $= \frac{\gamma\, sin\alpha d^3 \cdot tileDim}{6\, \mu \cdot tileDim^2} = \frac{\gamma\, sin\alpha d^3}{6\, \mu \cdot tileDim}$. It is important to note that this is a formula for laminar flow on a flat and smooth surface and will thus be less accurate for faster flows where turbulence and obstructions pose more of an obstacle[5].

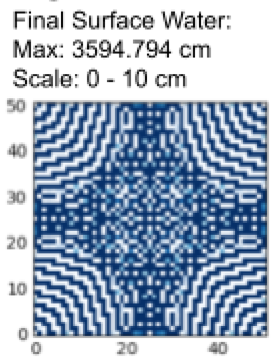
Since the $\frac{\gamma\, sin\alpha}{6\, \mu \cdot tileDim}$ is constant for constant elevation difference, it is able to be precalculated. This significantly cuts down the number of calculations being done inside the time loop. It is, however, forfeiting a bit of accuracy as calculating this way neglects the effect of dynamic water accumulation. There are several cases where this method of precalculation crucially overlooks changes caused by accumulated water on the flow. To get around this, my model has a checking segment that looks at the current water levels around it and searches for cases where the precalculated value is misrepresentative of the true flow (See Appendix A for more information). Through this segment, it retains the accuracies of dynamic flow while maintaining the significant speedup that precalculation brings. In the results of identical simulations -- one with dynamic flow and one without (Figure 4 -- for more clarification on reading these outputs see Appendix B) -- it is clear the impact that dynamic flow has on bringing more accurate results (look especially at the maximum depth of water in the non-dynamic flow results and areas around that seem to have lost their water despite being so close to the minimum where water should be collected).

Figure 4



However, this approach is not without its drawbacks. Due to the ability of water in this way of calculating to travel forward and then back, the simulation allows 'waves' to form. These 'waves' aren't natural waves that form on disturbed fluid surfaces but rather oscillations due to instability in time iteration. Figure 5 is a calculation run on the same elevation as Figure 4, but waves have formed and ruined the output quite absolutely

Figure 5



-- it is now in no way representative of what the water would do. The waves are caused when the time step size is too large: water that is flowing down a slope with high time step size will send almost all of its water to the tile beneath it. This begins to overfill tiles who then let all their flow out similarly. Figures 6.1-7 provide the steps of the formation of one of these waves. The input from the right stops step 4, and repetition is seen through steps 5 and 7. These waves only form at conditions of high water flow rates and relatively small change in elevation across grid squares (such that the water can easily surmount the change). The former of which can be caused by high depth of rain or high time step size. While the amount of rain that falls or the quantity of flattish regions isn't changeable, the time step size can be adjusted to prevent the formation of waves.

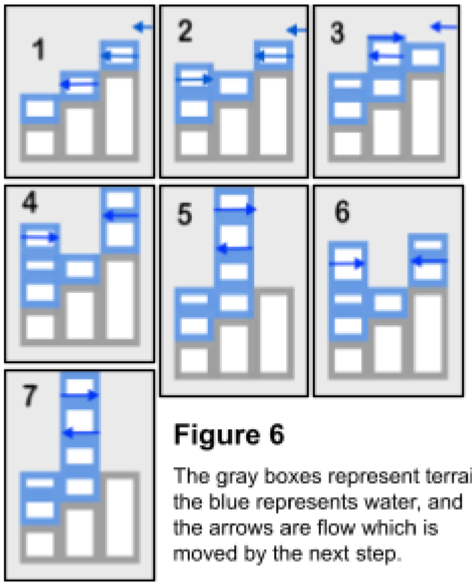


Figure 6

The gray boxes represent terrain, the blue represents water, and the arrows are flow which is moved by the next step.

Another crucial detail of my model is the use of a delayed write to the water arrays. The depth of water in the tile that is flowing outward and the depth of the water in the tile receiving it are both important to accuracy in the aforementioned checks. Thus, my model maintains separation between the data being used to make the calculations and the resulting changes to the data to ensure that water flowing into a square is not allowed to flow out of that square in the same time step.

Finally, erosion problems are marked from flows that exceed 40 cm/s which is the speed that erodes .01 mm silt to 3.5mm gravel[13]. The soils around Albuquerque are mainly between these sizes (sands fit right in the middle of that size range).

Verification and Validation -------------

There are three tests that make up the Verification and Validation of my model. Two are done on mathematical geometries to be sure of the logical flow of water in my model. The final test is done on real elevation and absorption data for Albuquerque, testing a recorded flash flood scenario[7]. That final test will be to assure the true accuracy of my model in predicting the results of a downpour.

To dive into the physical accuracies of my model, I will focus on the fluid dynamics and absorption will be disabled for these two verification tests. To the right is the flat plane defined by $z = .2x + .2y$ (See Figure 7.1) over a 5x5 grid. The area to fill the shape to z = .8 meters (See Figure 7.2) can be calculated with $V = \frac{1}{3}Bh$ with height h = .8 and the base B $= \frac{4 \cdot 4}{2} = 8$, so $V \approx 2.13\text{m}^3$. Thus, to fill this area exactly with rain over a 5m by 5m area, 0.0853m or 8.53cm would need to be dropped. Plugging all of that into my model for 5 divisions (5 grid tiles per edge), the results can be seen in Figure 8.

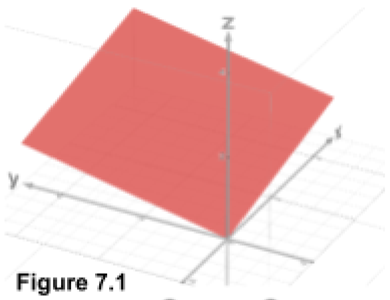
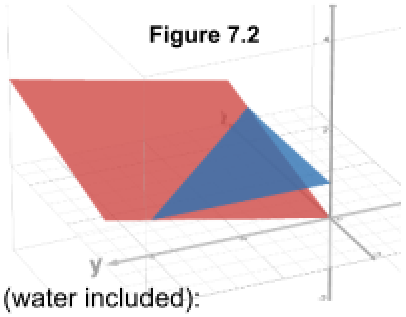


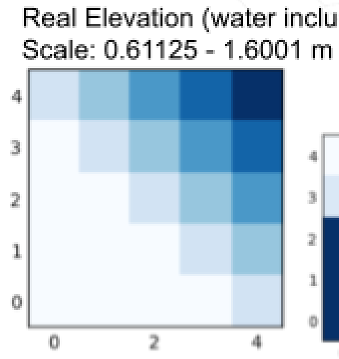
Figure 7.1

$z = .2x + .2y$

Figure 7.2



Seen on the real elevation plot, the elevations compare fairly well to the elevations in Figure 7.2. There is a discrepancy between the real water height at the end and what was calculated with the true triangular pyramid volume formula due to the finite number of grid squares. The grid approximation will always add a bit of error that really can only be cut down with more tiles (As shown in Figure 9 where the sides are divided up into 50 pieces rather than 5). In it, one can see how the water much more closely covers its 32% of the area and ⅘ of the adjacent sides, and how the water also quite evenly spreads out, making the bottom left of the real elevation almost all the same color (meaning that it is flat -- accurate to the flat surface of the pool of water in Figure 7.2). Increasing the number of squares makes this approximation a lot better. It is very important, too, to see that the expected maximum value in the corner is approaching the desired 80 cm, as the elevation approximation gets better. The most important



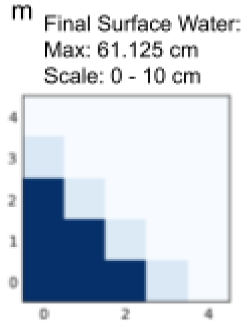
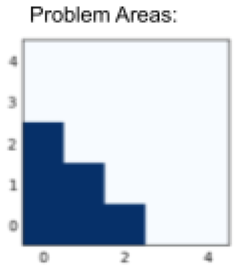
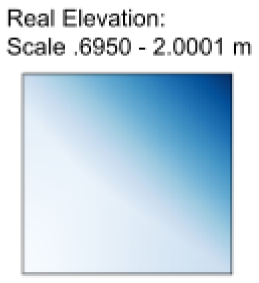


Figure 8
We expect to see 80cm as the maximum depth from the analytical tools for this geometry if the grid mesh perfectly approximated the elevation. 61.125 cm is expected for such a large grid pixel size.





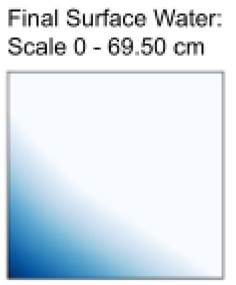
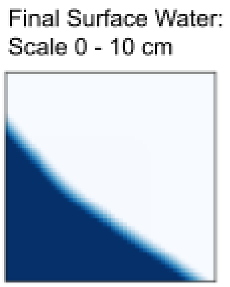


Figure 9
The maximum depth of 69.5 cm in this simulation is a lot closer to the desired 80 cm.



element of these results, however, is the settling of the water cleanly and evenly into the corner. This shows much of the power of the dynamic corrections -- without them all the water would flow into the corner with no heed to the fact that it had surpassed the level of the tiles around it.

A good check for the simulation in Figure 8 is to ensure that all the water that was input is accounted for at the end. Since the grid cells are 1m by 1m the volume of water in a cell is simply $depth \cdot m^2$. Thus, by calculating the sum of each square, the final volume is 2.1325m^3. That is very close to the actual input of 2.13m^3. The bit of error is easily attributed to rounding etc. as rain values are divided up not only across the terrain but additionally for a smaller time step.

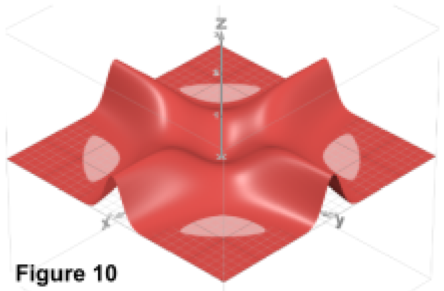


Figure 10

$$f(x,y) = \left(e^{-(x^2y^2)} - e^{-(x^2+y^2)}\right)$$

Another great example which has already been shown in Figure 4 and 5 is the mathematical terrain to the right (Figure 10). It boasts quite the wide variety of features: steep areas, flat areas, and even a cup of sorts. This test is a bit more complex. To test my simulation's ability to handle these three features, I have calculated the volume that the 'cup' shape can hold and a minimum rainfall to fill it. The model runs with these inputs to give the logical progression of water through the terrain.

First the volume: the edge is shortest at $\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$; thus, the volume is given by:

$$\int_{-\frac{1}{\sqrt{2}}}^{\frac{1}{\sqrt{2}}}\left(\int_{-\frac{1}{\sqrt{2}}}^{\frac{1}{\sqrt{2}}}\left(f\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)-f(x,y)\right)dy\right)dx$$

(Yes, surprisingly the "cup" is square)

Which evaluates to .339m^3 over that square of area ½m^2. Which, in turn, gives 0.6780 m or 67.80 cm of rainfall needed to fill the cup. Plugging that into my model, Figure 11 shows the result. The real elevation graph shows the inside of the cup is almost entirely evenly distributed and similarly with the flat areas outside the cup. It is crucial to notice the scale for the real elevation and how the slight differences in those flat areas really accounts only for small changes.

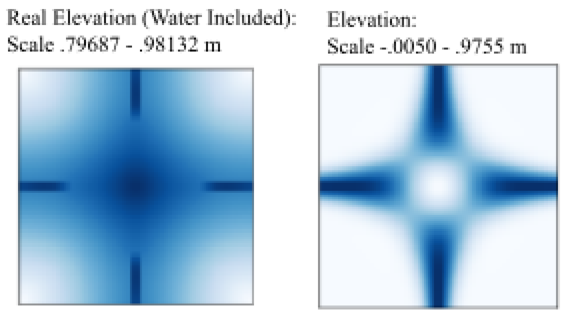


Figure 11

Across these two mathematical models, my simulation demonstrates the logical flow of water on these surfaces to the desired degree of accuracy. With these results so close to their true mathematical identities, my model can be certain to make accurate predictions regarding flow and accumulation.

Model geometries are nice to verify performance but can't really be used for any kind of practical predictions. While tested and initially verified on simplistic mathematical terrain, my simulation is meant to model real and important topographies. For the following validation simulation absorption is turned back on. To validate the true capabilities of my program (including absorption and flow with respect to time), the flash flood in the Embudo Arroyo will be simulated and compared to the actual rainfall accumulation recorded. Albuquerque Authorities[7] state that the foothills got more than an inch of rain in less than 15 minutes. The resulting flow in the North Diversion channel was 6 feet high. For this scenario, my model runs with 2.7 cm of rainfall over 15 minutes. Here is the prediction of the resulting water depth (Figure 12.1) with colors indicating water depths 0-10 cm (the maximum depth in this data set is 5163.6 cm), and data was taken 45 min after the rain stopped (data accompanied by corresponding elevation (Figure 12.2) and satellite imaging map (Figure 12.3)):

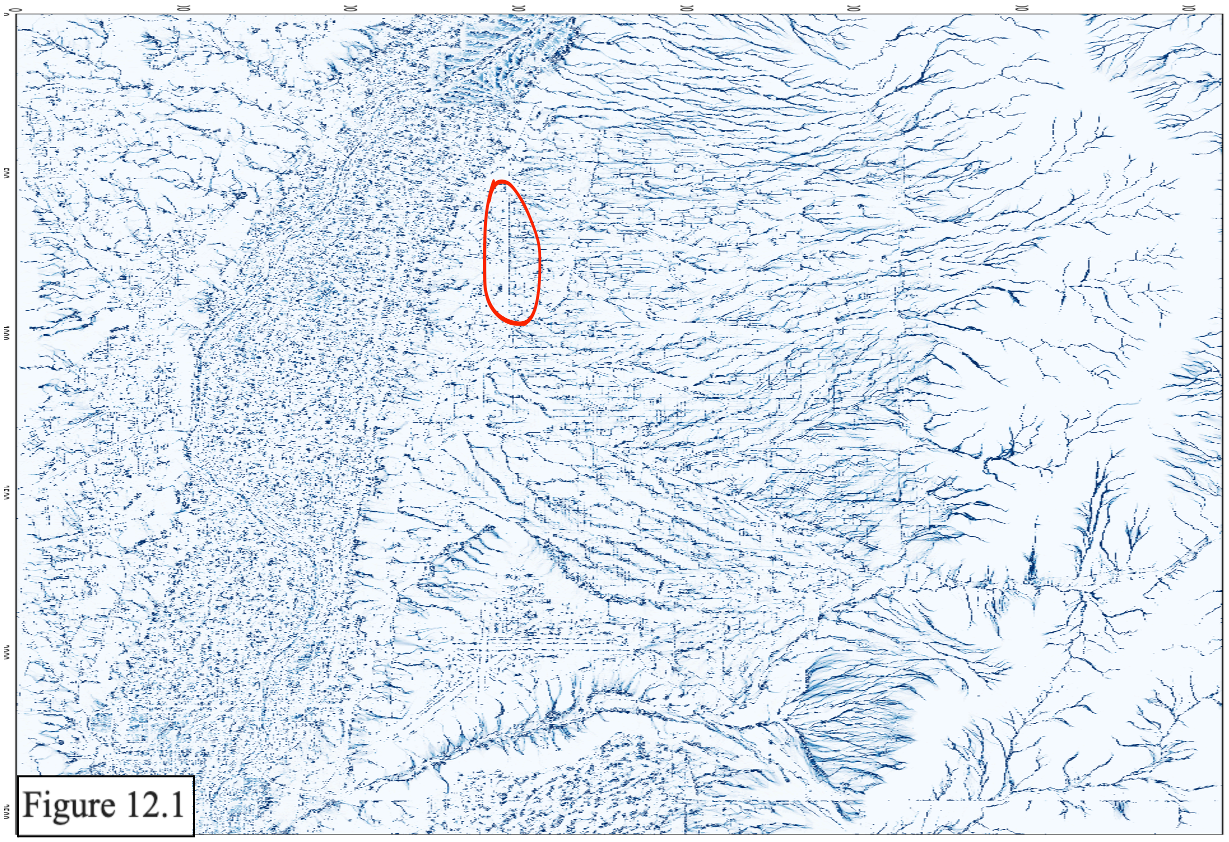


Figure 12.1

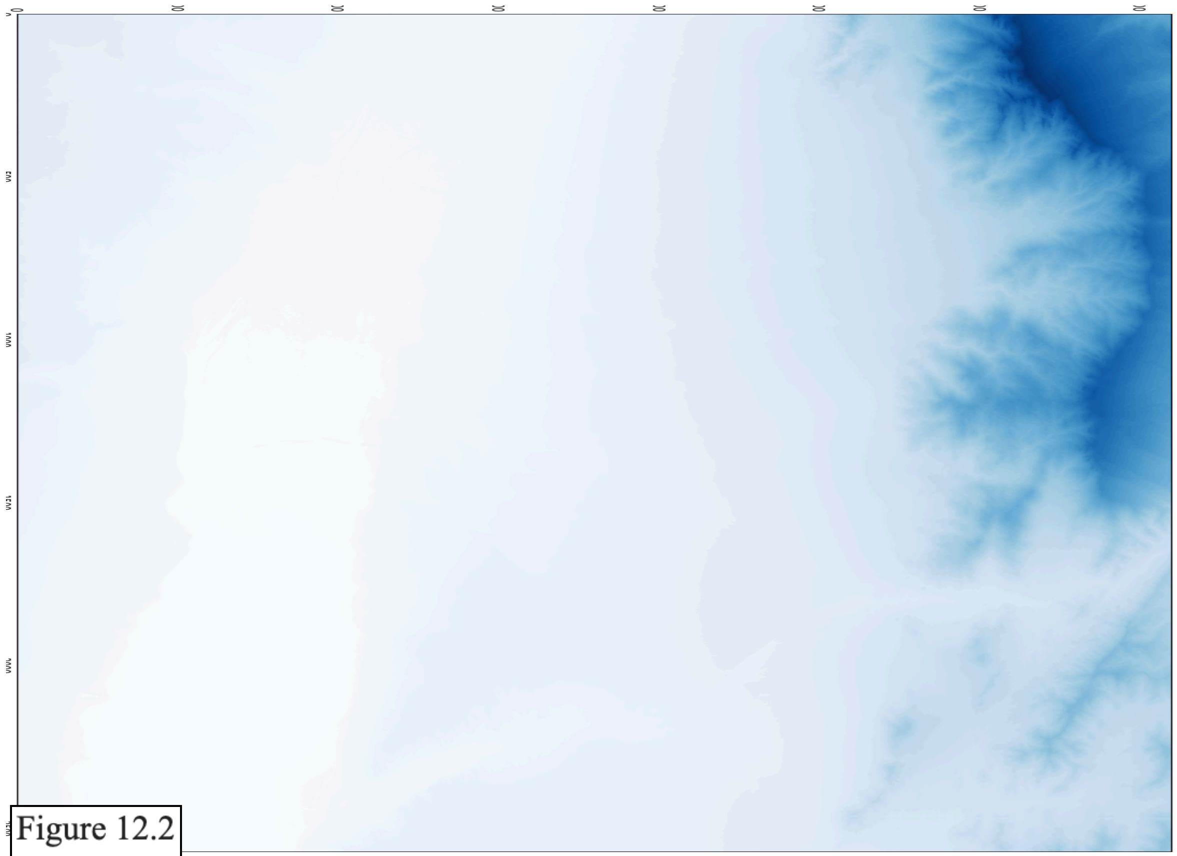


Figure 12.2

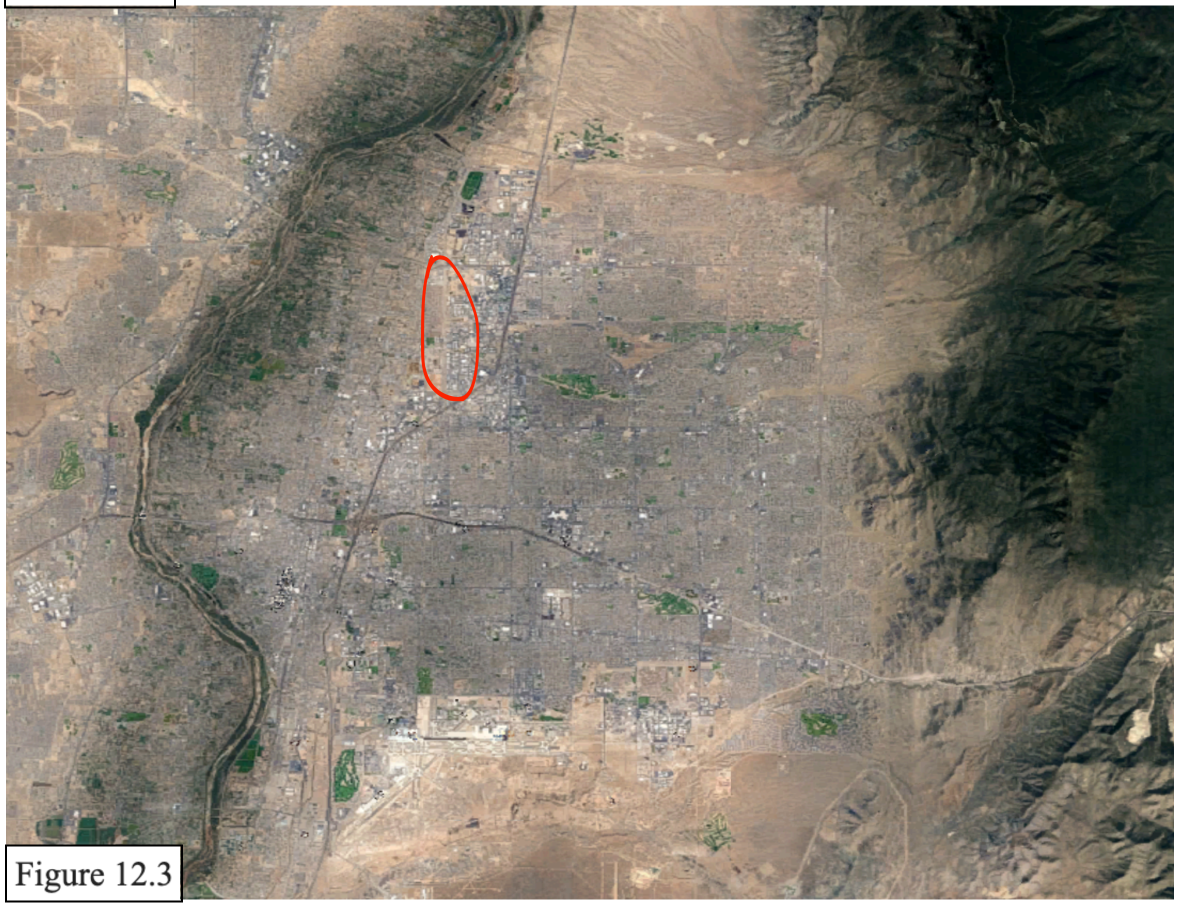


Figure 12.3

Using the reference map we can find the North Diversion Channel fairly easily (the vertical line circled in red on the water and satellite maps). Zooming in (Figure 13.1), there is already an important problem: there are the trademark patterns of waves from too large a timestep. Waves, when limited to small areas like this, can slow down flow rates to an extent and potentially cause water to flow over things that it normally wouldn't. It shouldn't affect the amount of water in an area too much. Zooming in further (Figure 13.2), it can be seen that there are about two pixels that make up the channel and the values of the filled squares are quite varying. To achieve a prediction in terms of depth in the real North Diversion Channel, the water present must be put over the true channel geometry. The average depth of water from my model is (64.95cm or 2 ft 1.98 in). When put over the new geometry (from a 24.60m linear strip to a channel of depth ~12m and sloping sides that make up 14.1m margins with a 7.8m flat bottom[10](approximations from Google Maps, See Figure 14)), it yields 1.64m or 5.47ft. 5.47ft is within 5.15% of the real result of 6ft. The minor deficit reported by my model can be attributed to the formation of waves in the simulation or the vagueness of the information about this validation scenario with respect to the time the flood waters reached 6 feet.

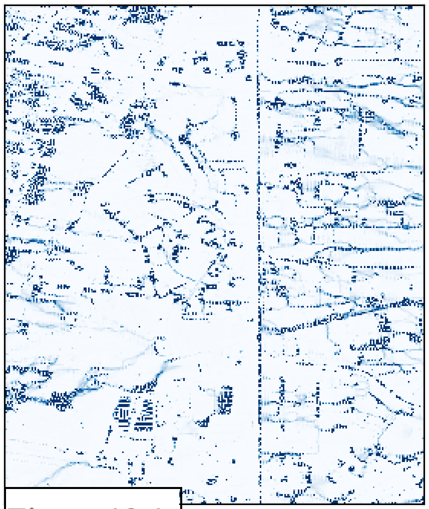


Figure 13.1

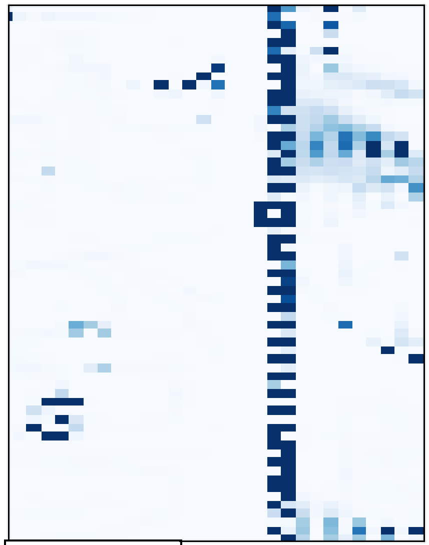


Figure 13.2

In any event, the results obtained from even a simulation with wave formations were quite accurate. With this accuracy for this problem, it is safe to use my model for true predictions or analysis of the current water management systems.

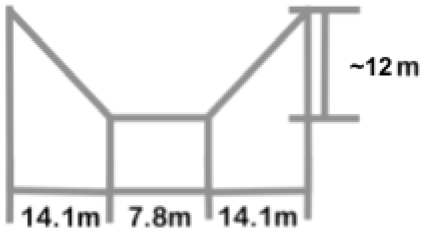


Figure 14

Results: Data and Analysis --------------------

Running simulations of more generic storms over the Albuquerque Area exposes some significant weaknesses in the water management system. Here, with a storm with 1 centimeter of rainfall over 30 minutes (data was taken right after the rain stopped), it can be seen that water is

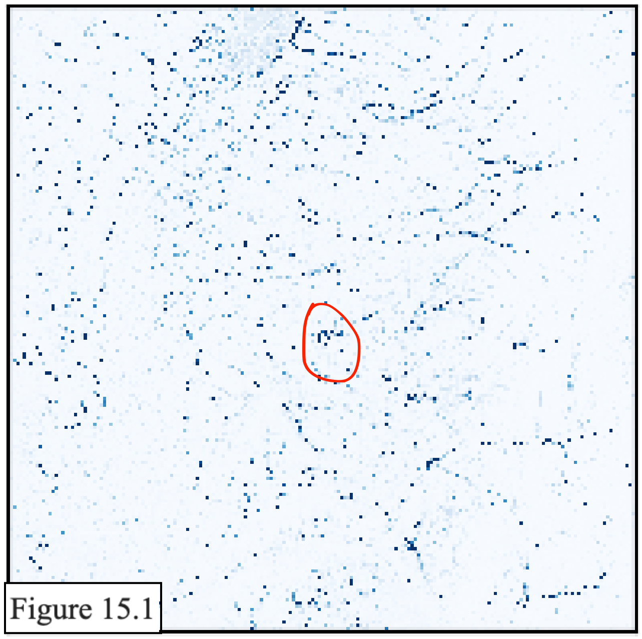


Figure 15.1

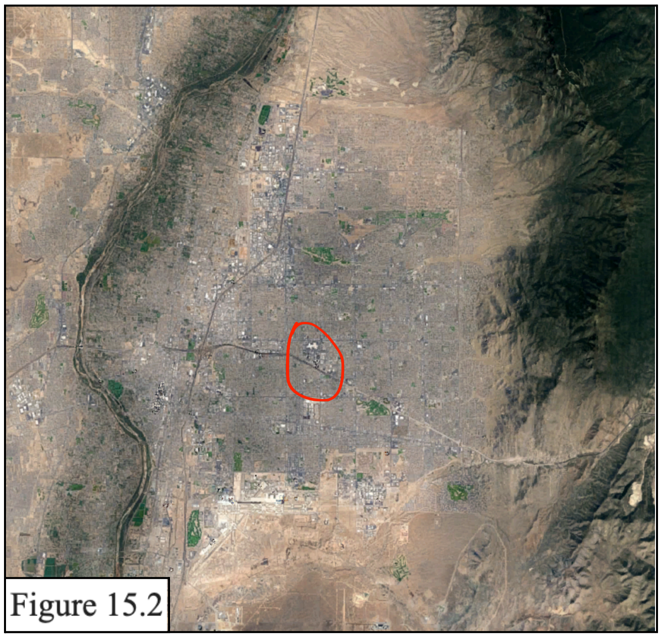


Figure 15.2

building up over and around the Ebudo Arroyo a little before where it meets 1-40. The first plot is final surface water with scale 0-10 cm. The second is the corresponding satellite map. It looks as though the infrastructure a bit up the Ebudo Arroyo failed at capturing these floodwaters and they ended up running through the surrounding neighborhoods, potentially causing significant damage. This is fairly unsurprising because the validation scenario was specifically recorded due to 3 deaths of people being swept away by flash flood waters in that area. It is evident that this area especially would benefit from redesign and improvement to be safe for the Snow Heights neighborhood through which it passes.

Another important detail of this simulation is the flood waters accumulating on the northern stretch of the Rio Grande Floodplain (the following data was taken 30 min after the rain stopped -- Figure 16.1: the area in question is circled in red). The first plot is final surface water with scale 0-10 cm. The second is the corresponding satellite map.

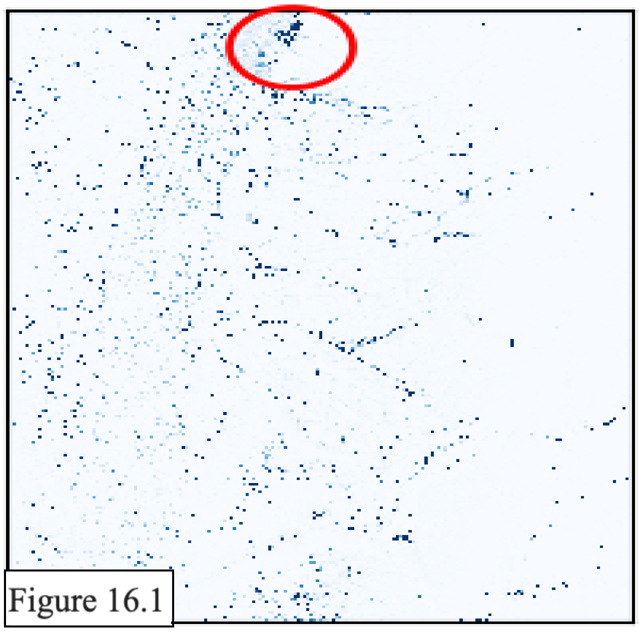


Figure 16.1

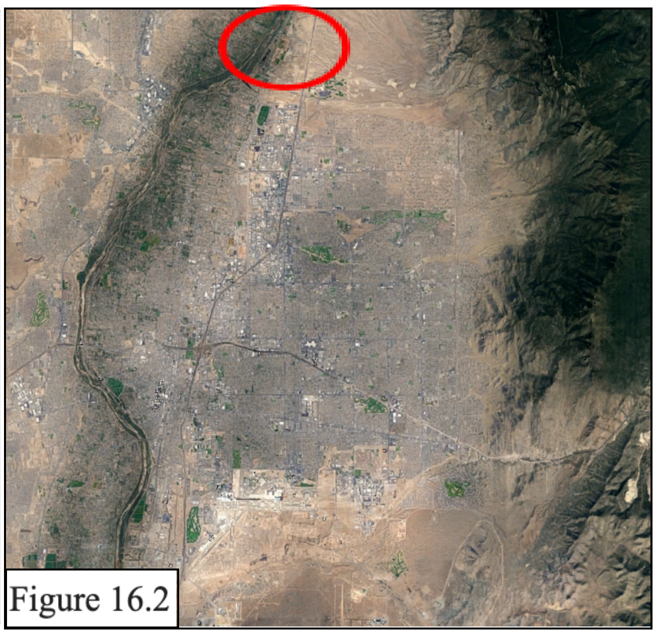


Figure 16.2

A lot of water is accumulating on the diversion routes to the east of the river and ending up on the farming/ranching lands close to the river. The water management infrastructure ends as it comes up on those lands. As seen below (Figure 16.3), the North Diversion Channel's outflow (circled in red) into the Rio Grande marks the last significant water diversion construction to the north. There are a couple smaller arroyos and two detention basins (marked in yellow); however, beyond the blue line there is really no more water diversion infrastructure[10]: only farms and some residential areas. The final step to get this water safely to the river is missing, and it is clearly causing some significant problems for these areas as seen in my model. It is crucial that this lack of water diversion infrastructure is addressed to protect both the agricultural ventures in this area and the people living there. It is also important to note that this area is also inhabited by a poorer part of the population, making them

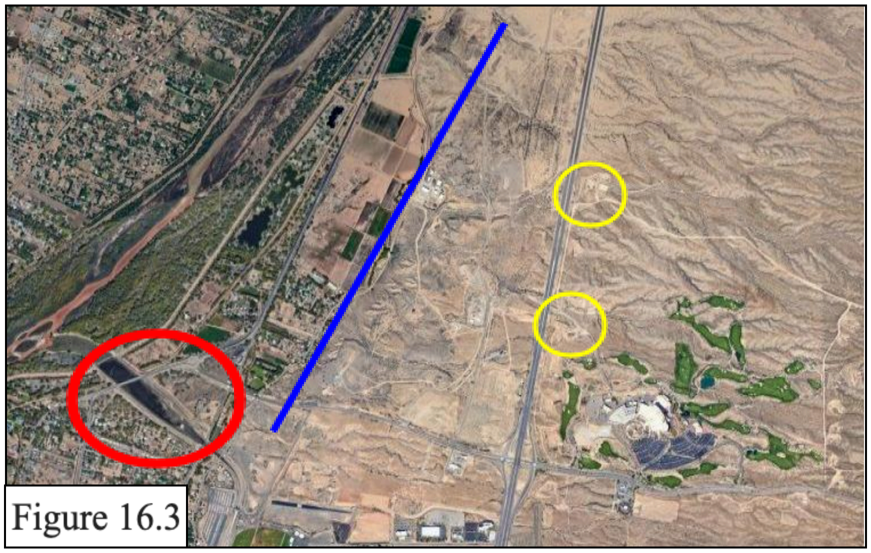


Figure 16.3

much more vulnerable to property damages and other consequences of uncontrolled flood waters.

A final important issue to address in the Albuquerque Area is erosion. With the desert's relatively poor quality soil, many of the more mountainous areas of the Albuquerque area are at significant risk of erosion. In a simulation of a storm with 1 cm of rainfall over 30 minutes, the following graph of Erosion Problems resulted (Figure 17.1 with corresponding satellite map: Figure 17.2).

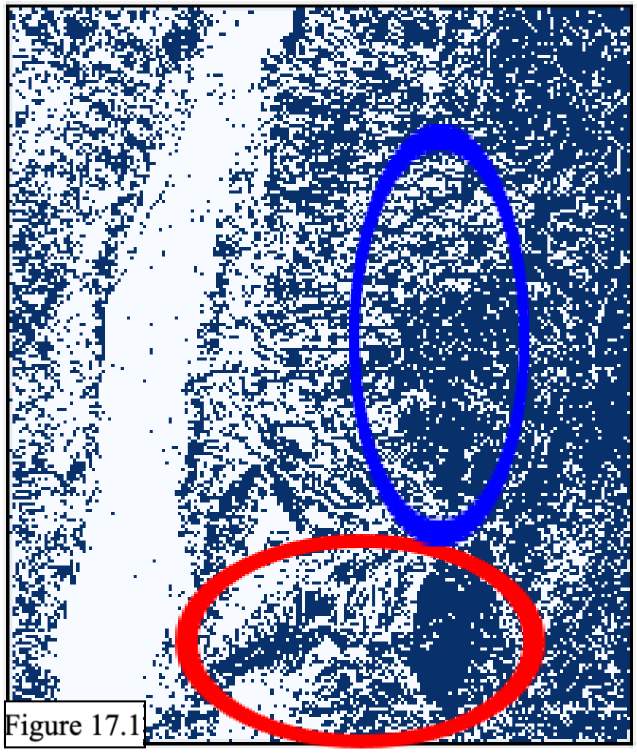


Figure 17.1

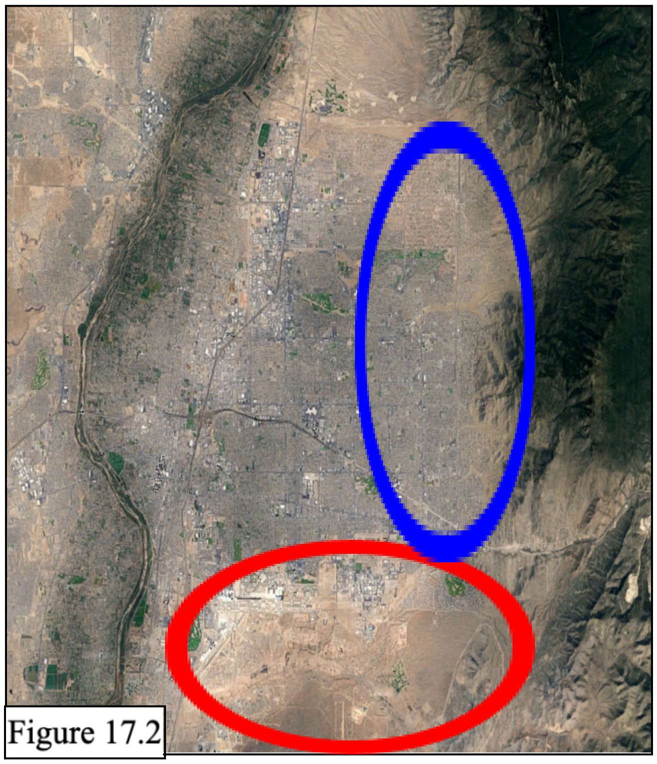


Figure 17.2

This Erosion Problems data marks tiles with a flow velocity greater than 67cm/s (slightly different than normal) which corresponds to .007 mm - 6.5 mm particles eroded[13]. This data says that there are high enough water flow rates to erode soil across the city and more prominently in the foothills and mountains (look especially at the Tijeras Arroyo and Watershed (circled in red)). It is important to note that this erosion warning does not differentiate between whether or not the soil particles are within the size range described above. For many neighborhoods in Albuquerque, concrete and xeriscaping dominate. These materials are not eroded so easily. It is important, however, to see the areas where that isn't the case. A prominent example of this is the aforementioned Tijeras Arroyo. For a good portion of its length, it is not reinforced with concrete, rather cutting through the desert (Tijeras arroyo has a soil bed for the entire length of the blue line in Figure 17.3, only switching to concrete after 1-25).

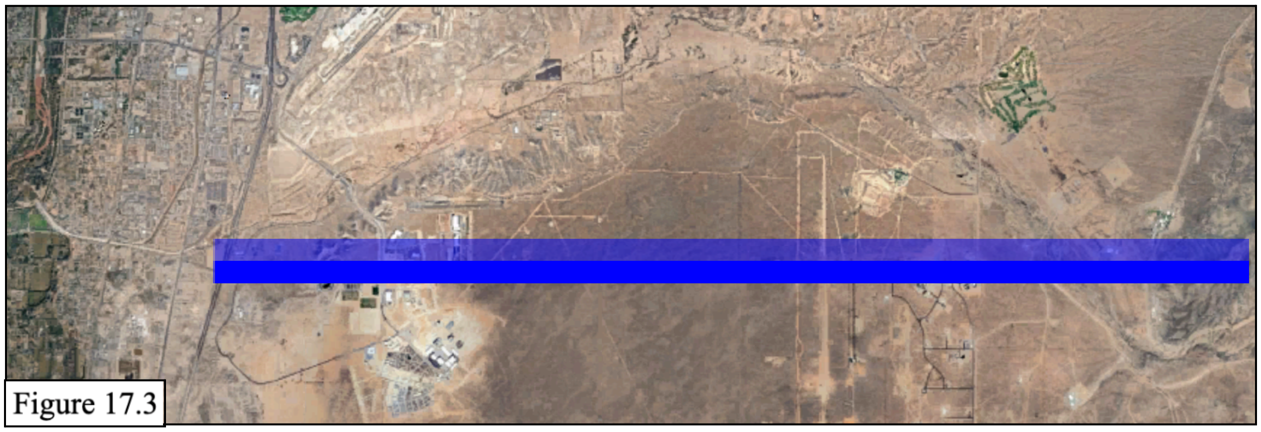


Figure 17.3

A storm as simulated by my model would simply exacerbate the condition of the Tijeras Arroyo upstream and potentially flow large amounts of sediment into water management infrastructure closer to the river. Much of the older water management infrastructure in the Albuquerque Area is similarly earthen and susceptible to heavy rainfall and flow to cause erosion. This infrastructure would clearly benefit from improvement and reinforcement to better prevent erosion.

Another significant area under threat of erosion is the foothills (circled in blue on Figure 17.1,2). The majority of residences in the foothills don't boast extreme vegetation or other changes that would protect them from erosion. Additionally, many of the hiking trails going into the mountains from there have clear evidence of significant erosion in many of the small gullies. The area doesn't have the best soil quality[8] which causes a cascade of problems: little significant vegetation to hold the soil together, often sandy soils that erode easily, etc. More detention basins and other flow slowing methods would help this area greatly and also get more of the runoff from the mountains absorbed into the aquifer to also benefit people and vegetation in those areas.

Finally, to address the high coloring in both the city and in the mountains, the more western city does require some attention with regard to erosion of loose materials, but due to large amounts of road and pavement that can't be eroded under these conditions, the city most likely does not need significant change and redesign regarding erosion prevention. As for the mountains, the mountains are mainly rocky, but the soil there supports much more significant vegetation, from grasses, to bushes, to trees. These all prevent soil in the mountains from being eroded, and in doing so, protect much of the watershed below. The mountain, despite its seeming susceptibility to erosion, is better protected.

Products: A Tool for Prediction ---------------

My model wasn't purely made for the examination of the Albuquerque Area and its local water infrastructure; it was made to predict precipitation results anywhere there is a storm and sufficient data to work off of. Coupled with the US NOAA, my simulation could add to their predictive weather services with results of the precipitation on local watersheds and artificial infrastructure. Besides my model and some fairly simple input and output data processing, there are no other products.

Given more time, I would like to reformat my code to run in parallel such that it would be able to run complex simulations much more quickly. Waiting for results was a significant obstacle during this process, and to truly use my simulation alongside NOAA as described above, it would need to be considerably faster. A generic simulation of flow over the Albuquerque Area takes around a day and a half to calculate, while they can get as lengthy as 3 days depending on the timescale the simulation is focused on. I would also like to try and impose more limitations on the formation of waves and try to root out ways to prevent them from occurring.

Personally, I think my greatest accomplishment of this journey was getting the precalculated and dynamic flow algorithm figured out, especially with respect to the unstable time iteration in 2d space.

Bibliography and Acknowledgements -------

1- Douville, H., K. Raghavan, J. Renwick, R.P. Allan, P.A. Arias, M. Barlow, R. Cerezo-Mota, A. Cherchi, T.Y. Gan, J. Gergis, D. Jiang, A. Khan, W. Pokam Mba, D. Rosenfeld, J. Tierney, and O. Zolina, 2021: Water Cycle Changes. In Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Masson-Delmotte, V., P. Zhai, A. Pirani, S.L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M.I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T.K. Maycock, T. Waterfield, O. Yelekçi, R. Yu, and B. Zhou (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 1055–1210, doi: 10.1017/9781009157896.010.

2 - "US EPA." *Climate Impacts on Agriculture and Food Supply | Climate Change Impacts | US EPA*, climatechange.chicago.gov/climate-impacts/climate-impacts-agriculture-and-food-supply. Accessed 31 Mar. 2024.

3 - US Department of Commerce, NOAA. "NWS ABQ Monsoon Awareness - Flash Floods." *National Weather Service*, NOAA's National Weather Service, 8 June 2018, www.weather.gov/abq/prepawaremonsoonflashfloods. Accessed 31 Mar. 2024.

4 - Flickinger, A.K., and Mitchell, A.C., 2020, Water-table elevation maps for 2008 and 2016 and water-table elevation changes in the aquifer system underlying eastern Albuquerque, New Mexico: U.S. Geological Survey Open-File Report 2020–1036, 9 p., https://doi.org/10.3133/ofr20201036. Accessed 31 Mar. 2024.

5 - "Chapter 4 Flow in Channels." *Chapter 4: Flow in Channels*, MIT OpenCourseWare, ocw.mit.edu/courses/12-090-introduction-to-fluid-motions-sediment-transport-and-current-generated-sedimentary-structures-fall-2006/72c0d3ee9656652db67315787b47c7a8_ch4.pdf. Accessed 14 Mar. 2024.

6 - Edge, Engineers. "Water - Density Viscosity Specific Weight." *Engineers Edge - Engineering, Design and Manufacturing Solutions*, www.engineersedge.com/physics/water__density_viscosity_specific_weight_13146.htm. Accessed 14 Mar. 2024.

7 - Reisen, Matthew. "AFR Recovers Third Body from Diversion Channel." *Albuquerque Journal*, 22 July 2021, www.abqjournal.com/news/local/afr-recovers-third-body-from-diversion-channel/article_2386f8eb-2f2c-5c10-9ea6-5e0808900963.html. Accessed 31 Mar. 2024.

8 - *Web Soil Survey*, websoilsurvey.nrcs.usda.gov/app/WebSoilSurvey.aspx. Accessed 31 Mar. 2024.

9 - Drohan, Patrick. *Understanding Hydrologic Soil Groups*, Villanova University / Value Engineering Inc., www1.villanova.edu/dam/villanova/engineering/VUSP/2019Sympresentations/1b/Dadio.pdf. Accessed 31 Mar. 2024.

10 - *Google Maps*, Google, www.google.com/maps/@35.1507816,-106.605217,71m/data=!3m1!1e3!5m1!1e4?entry=ttu. Accessed 3 Apr. 2024.

11 - "Flood Impact." *Protective Actions Research*, US FEMA, community.fema.gov/ProtectiveActions/s/article/Flood-Impact. Accessed 8 Apr. 2024.

12 - *TNM Download V2*, apps.nationalmap.gov/downloader/. Accessed 31 Mar. 2024.

13 - Earle, Steven. "13.3: Stream Erosion and Deposition." *Geosciences LibreTexts*, Libretexts, 6 May 2022, geo.libretexts.org/Bookshelves/Geology/Physical_Geology_(Earle)/13%3A_Streams_and_Floods/13.03%3A_Stream_Erosion_and_Deposition.

Idowu, John, et al. "Soil Health-Importance, Assessment, and Management: New Mexico State University - Be Bold. Shape the Future." *Soil Health-Importance, Assessment, and Management | New Mexico State University - BE BOLD. Shape the Future.*, Dec. 2019, pubs.nmsu.edu/_circulars/CR694B/.

Plummer, L.N., Bexfield, L.M., Anderholm, S.K., Sanford, W.E., and Busenberg, Eurybiades, 2012, Geochemical characterization of ground-water flow in the Santa Fe Group aquifer system, Middle Rio Grande Basin, New Mexico (ver. 1.2, November 20, 2012): U.S. Geological Survey Water-Resources Investigations Report 03–4131, 395 p., CD-ROM in pocket. (Also available at https://pubs.usgs.gov/wri/wri034131/.)

"TOPMODEL Calculation." *Hydrology - USU*, Utah State University, hydrology.usu.edu/rrp/ch6resources/tm.htm. Accessed 31 Mar. 2024.

Acknowledgements

I would like to acknowledge the inspiration I got from the TOPMODEL Calculation presented by Utah State University. I also significantly benefited from the guidance of Drew Einhorn, the scientist who reviewed my interim report and who laid out much of the journey to getting my program up and running.

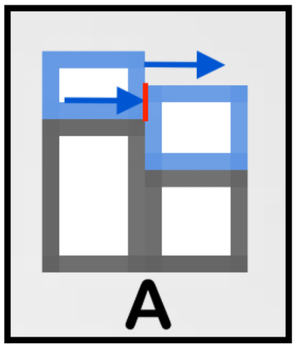
Both of my parents were also incredibly helpful throughout the process. My dad, a computer scientist for Sandia National Labs, helped me to take a deeper look into my code and understand the process (and what was going wrong). My mom, fluid dynamics/polymer material sciences researcher for Sandia National Labs, recommended the fluid dynamics flow equation for more accurate flow predictions and also helped me to take a step back when my code was not cooperating. My mom also helped greatly with reviewing and revising this report.

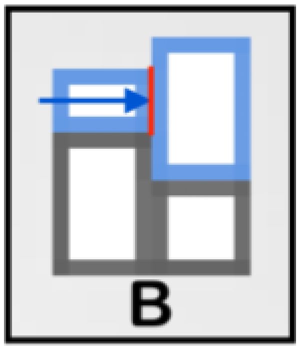
Finally, I would like to thank both Jonathan Wheeler and Karl Benedict at the UNM Library for their help getting absorption data and normalization tools for helping with compatibility, both of which I was unable to find on my own.

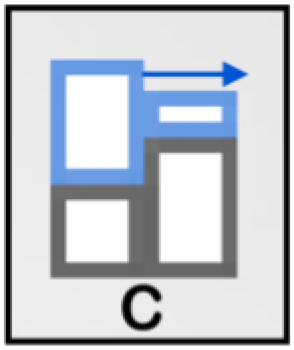
Appendix A ------------------------------------

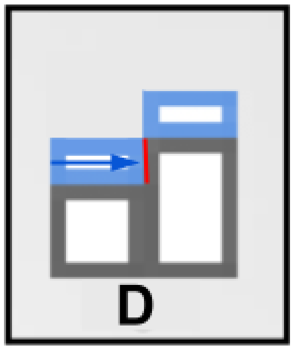
This Appendix concerns precalculation which is the calculation of the constant piece of the fluid dynamics equation used for prediction water flow (See page 5,6 for more information on the flow equation). However, precalculation can be inaccurate when water begins to change what precalculation assumes to be constant elevations.

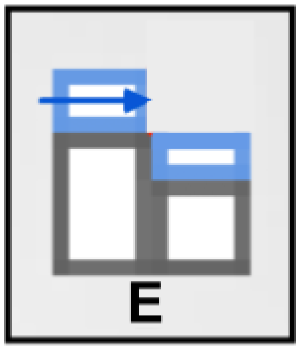
There are 6 cases in total for flow regarding water levels and elevations. There are three cases of those six where precalculation fails. For the examples below, they are viewed with increasing altitude being the top of the figure, the gray pieces represent terrain elevation, the blue pieces represent water accumulation on top of the terrain, the arrows represent flow, and the red lines are where flow tries but should not go. For these cases water flow will only be considered from the left tile to the right tile. My model only considers flow this way, as it iterates through all of the tiles and considers each potential flow one sided-ly. For case A, the precalculation says flow is all good to go; however, in

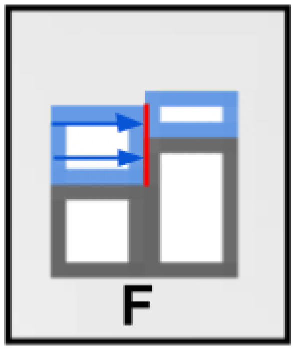










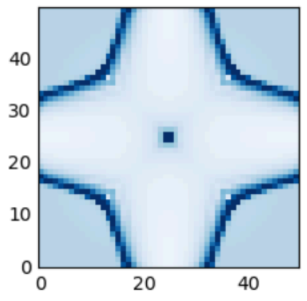


reality, the flow is impeded by previous accumulation of water on the right tile. Thus, case A needs dynamic recalculation. Case B is similar. Precalculation says that flow is good, but really, no flow can take place at all due to the fact that water wouldn't flow 'uphill' to the top of that higher water level. Since reality does not agree with the precalculation, dynamic recalculation is necessary. Case C is essentially the inverse of B. Precalculation says that no water should flow because the right tile is uphill. However, the water has reached high enough accumulation that it will flow into its neighbor -- recalculation is needed. Case D is different. Precalculation says no flow, and reality agrees. No recalculation is needed. Case E, too, is cooperative. Precalculation says that there will be unimpeded flow and reality agrees. Finally, Case F is a bit of an edge case. Precalculation gives no flow, and while there is sufficient water accumulation on the right to surpass the terrain, the water accumulation on the right is still enough to prevent any flow from left to right, so reality also says no flow. It ends up not needing recalculation. In Cases A, B and C, the precalculation overlooks an important way water needs to be handled in the model. Those cases are fairly easily defined which allows the program to go through a bunch of quite fast if statements rather than getting bogged down in calculating and comparing for every tile.

Appendix B -----------------------------------

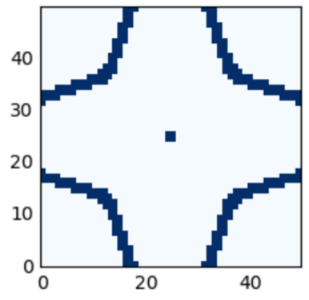
Understanding the outputs of my model is crucial to understanding its capabilities and my analysis of it. I will run through what to expect from each named dataset here. First and most common is "Final Surface Water". Final Surface Water data sets come in two varieties: scale of 0-10 cm and scale of 0-max. The first has the colors on the plot corresponding to the value as it is on the scale 0-10 (numbers between 0 and 10 get assigned a corresponding color intensity based on where it is on the scale, whereas numbers above 10 get assigned the same color intensity as 10 cm, and similarly with values below 0 and the color intensity for 0). The 0-max variety of graph is the same however the maximum value of the data set is made to be the upper bound for graphing. Graphing over 0-10 cm is useful when the details are really in what is in that range 0-10, especially when the max is in the thousands and graphing with the range as the max makes everything else colorless by comparison. Often for graphs over the scale 0-10 cm, the maximum value will also be provided as a point of reference. The actual data on this graph indicates the depth of water predicted at the points on the graph.

Final Surface Water:
Max: 102.44277785402164 cm
Scale: 0-10 cm



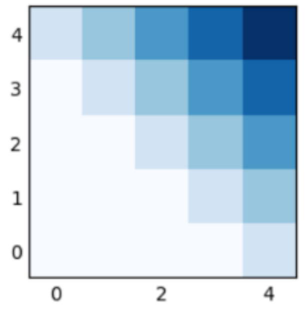
The next type of graph is the "Problem Areas" and "Erosion Problem Areas" data sets. These are provided with no scale and are simply interpreted as boolean graphs. Pixels with color are problem areas; pixels without color are not.

Problem Areas:



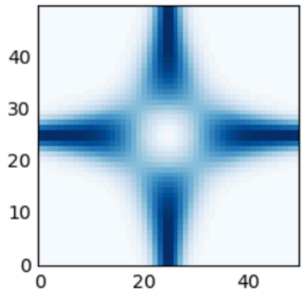
Next are "Real Elevation" data sets. These, much like "Final Surface Water" plots, will be accompanied by a scale to indicate the range on which it was graphed. These graphs will always be graphed from maximum to minimum value. The data on these sets is the water level (now in meters) added to the elevation to get a true idea of the water accumulation with respect to the terrain.

Real Elevation (water included):
Scale: 0.6112500559617084 - 1.6000646282673698 m



Final are "Elevation" plots. They are also always plotted from minimum value to maximum value. Their data is the elevation of the terrain at those points.

Elevation:
Scale: 3049.7452996079974 - 3098.776331460856 m



Computational Hydrodynamic Analysis for Speed Maximization (CHASM)

2024 Supercomputing Challenge

Luke Rand, Greta Swanson, Nandita Ganesan

Final Report

Fluid Mechanics
Santa Fe Preparatory School
Santa Fe, NM, US
April 10 2023

Contents

Introduction

1 Executive Summary

With access to adequate coaching becoming an ever increasing issue, disproportionately affecting underprivileged groups and individuals, the need for technological coaching aids to cut expenses increases. Swimming faces this challenge, and thus requires remedies. Our project, Computational Hydrodynamic Analysis for Speed Maximization (CHASM) achieves a method to provide coaching at low expense for swimmers without access, as well as increase the efficacy of pre-existing coaching programs for the sport of competitive swimming, potentially allowing the growth and increased accessibility of aforementioned teams. CHASM provides machine augmentation and replacement for human coaching time through computational analysis of the movements of the swimmer. Our model takes captured video footage of a swimmer, analyzes it using hydrodynamic drag approximation, and outputs a simple visual to allow either the swimmer themselves or a coach to locate points of human error. Integration of the hardware and model into swim teams and local pools can expand access to the sport of competitive swimming and address inequality issues related to coaching access. Additionally, with many people around the world facing injuries or disabilities that limit their movement and change their bodies, individualized coaching for swimming, a commonly prescribed recovery sport, is more important than ever. CHASM addresses this as well, with direct individualization providing the ability to account for such differences.

2 Method Outline

- First, an inexpensive PVC frame is attached to the lane line on either side of a swimming lane. This is the Lane Videography Unit (LVU) and is used to capture footage of the swimmer from specific angles using Apple iPhones which are mounted in waterproof casings on the LVU. IPhones are used to capture video footage due to being a device many coaches and swimmers will already have, and thus do not need to buy.
- Next, a short script takes individual frames from the footage collected by the LVU at an inputted time, and passes them to the model, CHASM, to output data to reduce drag.
- First, CHASM uses a pretrained image model to develop two-dimensional wireframes from the two images associated with each angle.
- CHASM then combines these two-dimensional wireframes into a single three-dimensional wireframe.
- This wireframe is passed to a `Swimmer` object constructor.

- Included in the swimmer object is a function to calculate the approximate drag on the body represented by the wireframe. An optimization function uses the drag approximation function to determine the ideal wireframe for the swimmer object to minimize drag for the exact proportions and sizes of the swimmer.
- Now, both the original wireframe, derived from the swimmer's position, and the optimized wireframe, derived from drag calculations for the swimmer's body attributes, are visually displayed to the user.
- Either the user or a coach of the user can use the outputted data to correct errors and improve swimming performance. The CHASM model can then be used to verify progress and determine further steps towards improvement.

3 The Problem

Across the United States, access to swim coaching has been limited for a variety of reasons, one of the most pertinent reasons being cost. A private lesson with an instructor can cost on average one hundred to three hundred dollars per hour (Dougherty, 201) and most competitive swimmers take private lessons, on average, twice a week, with more dedicated swimmers taking well over two (Koury, 2024). This means a significant amount of money is dedicated to the sport from the swimmers family, and for many, the price is not worth the benefit (Dougherty, 201).

The majority of swimmers will instead join a club or school swim team instead as these have a much lesser price. However, a club team has, at least, upward of thirty athletes on average, and highschool swim teams have upward of forty plus on average, depending on the size of the highschool. Meanwhile, the number of coaches for these teams leads to an approximate ratio of 1:10, coaches to athletes respectively. This means that a coach's attention has to be split between several athletes and each athlete will not receive the attention of a coach in the same way that an athlete taking privates could (Here).

Our program helps to bridge the gap between these two issues. By having a program that is cost efficient and coaches students, it would allow students who are in a high school or club team to have easier and cheaper access to individualized coaching.

4 Current State of the Field

As the sport of swimming stands, technology plays a significant role. Electronic timing machines are used to collect results for races and heart rate monitors are used for training purposes (Wise, 2022). Existing companies provide side by side comparison of video footage and method to review details, such as STREAMLINE. However, human error and labor remains as a key crutch point, as coaches are still the ones analyzing and watching any collected data. Despite innovations in technology such as glass sided pools for video recording and analysis, CHASM remains unique in a computational approach to the analysis itself, and provides quantitative corrections based purely on the laws of hydrodynamics, which are immutable and infallible.

5 The Streamline Position

The streamline position significantly reduces the amount of drag acting on a swimmer during the dive and push off state of a race or swim, occuring at the beginning and all turns, respectively. This reduction in drag can be advantageous to many swimmers, as reducing drag allows for the swimmer to reach faster speeds (The Importance). For example, a swimmer can spend up to 20 percent of breaststroke in a streamline position, and by having a quicker streamline, a swimmer is able to drastically improve their times for competitions (Poirier, 2023). Beyond this, the streamline allows a swimmer to have a quicker break at the start of the race and carry that momentum and speed throughout their race. If a swimmer has a subpar streamline, they will lose speed at the beginning of their race and have to regain that speed later on, wasting energy (Admin, 2021). The streamline also occurs off the wall at every turn, and lost energy can add up over longer races (Poirier, 2023). Many beginner swimmers are able to improve their streamline and show drastic improvement in their streamline within the first year, as a tight streamline does not require any additional energy or work to generate *(fig. 1)*.

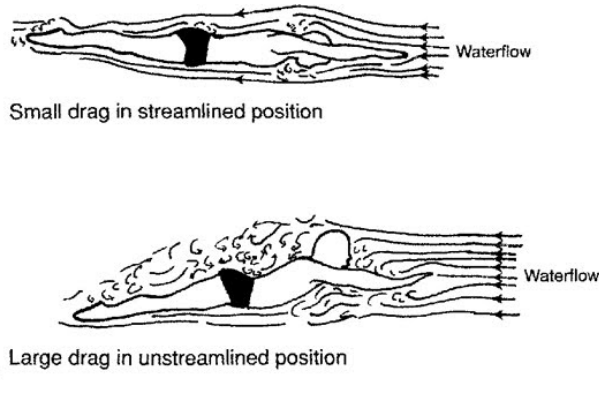

Waterflow
Small drag in streamlined position
Waterflow
Large drag in unstreamlined position

Figure 1: Streamline position of swimmers (360swim).

Beyond the importance of streamlines in competitive swimming, we were limited by our access to technology that would have allowed us to gain the data needed for strokes such as freestyle. Swimming strokes which involve adding energy to the system through movement are significantly harder computationally, and were not approachable within a year time frame. However, a streamline negated many of these concerns, facilitating ease of designing the apparatus and simplification of computation, as a part of swimming that only involves energy loss through drag and not energy gain and representing an important part of the sport.

6 Importance of Individualization

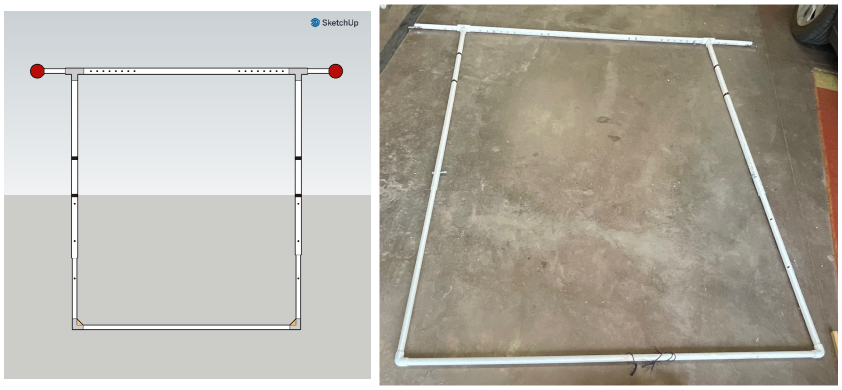
An ideal swimming streamline consists of arms forward, hands overlapped, and body pointed straight (Holmes, 2022). With clearly defined and well tested data on optimal swimming position for a streamline, individualized calculated data for an optimal position seems unnecessary. However, human bodies differ in size and proportion, and while general technique may remain constant, exact angles and position differ immensely from person to person, and can often only be picked up visually. Furthermore, body types and abilities can differ even more when injuries and disabilities are introduced. Our model is especially able to address these issues, visually displaying data specific to a person's body type and proportions, providing accurate coaching for anyone, and accounting for individuals with differences, whether temporary or permanent. Additionally, if our model were to be expanded for use past just the streamline and into the domain of other strokes such as freestyle, individual body type influences correct swimming form multitudes more, and the technology we outline would become even more pertinent.

Computational Analysis for Drag Optimization (CHASM)

7 Capturing and Isolating Frames for Analysis

7.1 The Lane Videography Unit

The capture of video is performed using a custom built Lane Videography Unit (LVU). The frame is PVC, certain fasteners are steel, and the iPhone holding mounts are wooden. PVC is chosen for the main frame for its ease of construction, lightweight nature, and relatively low expense. The entire LVU contains under fifty dollars of material. Figure 2a shows the design for the LVU. The vertical axis contain two nested PVC pieces with holes drilled through both pieces. A screw and washer are then able to go through a set of holes decided by the user, creating adaptability for different depths of streamline. The black lines, denoted by tape, demarcate the optimal height of the streamline, most directly in the center of the path of the cameras. The horizontal axis also contains similar adaptability to account for different widths of the lane. The cameras are mounted at 45 degree angles for ease of computation, using a square root of two constant for conversion to a three dimensional wireframe as opposed to calculated trigonometry values. The decision to put cameras on angles arose from the understanding that the diagonal distance is longer than the perpendicular and can more easily fit the entire person within the frame, given the constraints of the minimum width of a swim lane.



(a) (b)

Figure 2: (a) The LVU's blueprint (b) A completed frame of the LVU

7.2 Taking Individual Frames

Since video footage is taken, and not always started at the same time, a short python script determines the beginning of each video, is given a specific time from one of the videos, and adjusts a second time for the second video. A frame from both videos is then returned from the same world time, regardless of video start time.

8 Constructing a Three-Dimensional Wireframe

8.1 Clarifying Terminology

The CHASM model uses two two-dimensional wireframes of a human body to construct a single three-dimensional wireframe. The body of the swimmer passing through the water is horizontal and positioned perpendicular to the bottom of the pool. Between multiple wireframes and a body positioned horizontally, the terminology in respect to coordinate axis can prove unintuitive. Thus, in order to clearly describe position, it is necessary to clarify which axis each label refers to. For the purposes of this report, the Y axis will be used to describe the horizontal axis between a swimmer's feet and head, with lower numbers at the feet and higher numbers at the head. Similarly, the X axis runs from right to left (from the swimmer's perspective, meaning left to right from a viewer's perspective), and the Z axis runs from the swimmer's chest to back. X_1 and Z_1 are used to refer to the non-Y axis of the left and right two-dimensional wireframes, respectively.

8.2 OpenCV Pose Recognition

We performed two-dimensional pose recognition for two-dimensional wireframes from images using the OpenCV library and the MPII Human Pose Dataset (Andriluka et al., 2014). We use a pretrained model (Gupta, 2018) which integrates the MPII Dataset with the OpenCV image processing library to derive points at specific locations, which we use to construct our wireframe. Usage of the model consists of three parts. Initially, we load the model network to be used on our images. Two model networks must be initialized, one for each of the two angles in our image. Images are then rotated so the Y axis is vertical on the image. The MPII Dataset primarily includes upright bodies, meaning vertical rotation is important for accurate wireframe point location. Next, we determine the frames necessary for the model to find wireframe points. A frame is essentially OpenCV's interpretation of an image, and involves locating the images in a file structure and using a constructor to initialize each frame from the corresponding image. Lastly, the frames are given to the model, which outputs a two-dimensional wireframe for each frame *(fig. 3)*.

In order to use the wireframes computationally, it is necessary to make a few changes to each output wireframe. The location of the point corresponding to the chest is determined and its corresponding vector is subtracted from the vector of each wireframe point to "zero" the wireframe around the origin. The result is a translation of the wireframe to the chest point as the origin. This makes combination of wireframes significantly easier and improves compute time by minimizing the need for vector subtraction later on. Additionally, the wireframe is vertically inverted along the Y axis so that drag can later be computed from bottom to top, low Y values to higher numbers.

8.2.1 Code

```python
#Method to translate points for computability
def translate_points(points):
    #determine chest point and initialize new points
    chestpoint = points[14]
    new_points = []
    #adjust each point
    for i in range(len(points)):
        new_point = [0, 0]
        #adjust each axis
        for j in range(2):
        new_point[j] = points[i][j] - chestpoint[j]
    new_points.append(new_point)
  return new_points
```

8.3 Combining Wireframes to add Depth

As previously stated, the two-dimensional wireframes from each angle of video are taken from diagonals, this makes constructing a three-dimensional wireframe

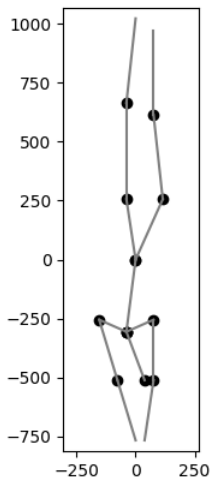


Figure 3: A two-dimensional wireframe determined from a streamlined swimmer.

mathematically more complex, yet reduces the margin of error and mitigates cost. Mathematical complexity results from converting the two diagonal coordinate systems geometrically into a single three-dimensional standard Cartesian coordinate system, but the diagonal approach is a better course than taking horizontal and vertical two-dimensional wireframes for two main reasons. First, diagonal perspectives on the 45 and 135 degrees ensure that all points are present in both angles, reducing the need for a third camera. Second, on a square frame, diagonals allow us to position the cameras further from the swimmer, mitigating perspective distortion, an image distortion phenomenon that intensifies near the edges of an image. Using the diagonals, we calculate the X, Y, and Z values of each point in three-dimensional Cartesian space from their two-dimensional counterparts using formulas determined geometrically. Due to the engineered 45 degree angle nature of the right triangles formed by the lines of view of the cameras, consistent algebra using a constant value of the square root of two can be used, significantly increasing performance times than what would be necessary for trigonometric computations. Each pair of corresponding two-dimensional points is used to calculate a single three-dimensional point, and each three-dimensional point is then appended onto an array to create a three-dimensional wireframe of the swimmer *(fig. 4)*.

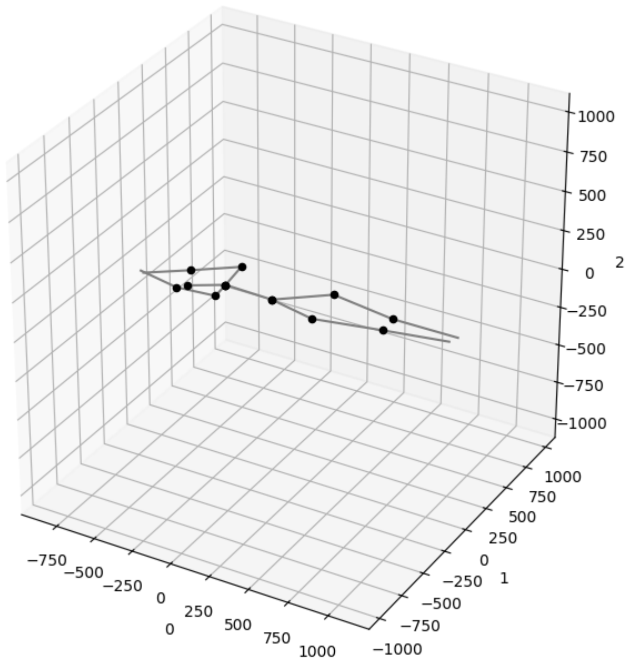


Figure 4: A three-dimensional wireframe generated from a two-dimensional wireframes.

8.3.1 Code

```
#method to calculate point in 3d from 2d points. cam1 and cam2
↪  are 2d points in [y,x,z] format
def getPoint(cam1, cam2):
    #argument parsing
    #x1 and z1 refer to the 2d image while x and z refer to the
    ↪  3d. Y is the same for both
    ys = [cam1[1], cam2[1]]
    x1 = cam1[0]
    z1 = cam2[0]
    #initialize point to return
    point = []
    #average y values for y
    point.append(sum(ys)/len(ys))
    #converting diagonals to horizontal and vertical coordinate
    ↪  system
    x = (z1+x1)/root2
    z = (z1-x1)/root2
    #append to point
    point.append(x)
    point.append(z)
    return point
```

9 Determining Drag Associated with a Wireframe

9.1 Determining Flow Type

Flow over a body can be characterized as one of two types: turbulent and laminar, with laminar flow characterized as separate layers with little interference, generally seen in denser, slower moving fluids, and turbulent flow characterized by high levels of interference, conversely with lighter and faster fluids. Drag created by a body moving with laminar flow along its edges is generated primarily by surface area drag, while turbulent flow creates form drag, produced by interfering layers of fluids (”Laminar and Turbulent,” 1992). The distinction between flow is based on a Reynolds number, calculated using the velocity of the body, the “characteristic length” of the body, which is the head to toe Y-axis length of the swimmer due to it’s relatively flat form, and the viscosity of the fluid (Ungerechts, 1982). For a body moving through fluid, a Reynolds number of below 1 indicates that flow is laminar, while a Reynolds number above roughly 10^6 indicates completely turbulent flow (OpenStax College, n.d.). The Reynolds number of a competitive number is in the interval $(2 * 10^5, 2 * 10^6)$, directly situated on the cusp of completely turbulent flow (Ungerechts, 1982). Therefore, our model must take into account both surface area drag and the interference of layers of fluid, commonly referred to as form drag. These two components become increasingly important in the creation of a drag approximation model, as it determines which forces contribute to total drag.

9.2 Drag Approximation Model

To determine drag, the force acting against the swimmer and the primary reason for energy loss in the streamline, we first determined constituent parts contributing to the drag force. As we determined based on the Reynolds number of a swimming body, drag acting on the swimmer will be frictional and form based, both in significant proportions (What is Drag?, 2022).

9.2.1 Approximating Skin Friction

Skin friction is directly proportional to surface area in contact with the fluid, in our case water, and can be calculated with the surface area equation for a geometric shape, and more complicated surface area determination techniques for more complex shapes. Since we do not yet know how much skin friction influences drag under our circumstances, we multiply the surface area by an arbitrary constant, A. This finalizes our skin friction calculation equation:

$$A * SurfaceArea$$

9.2.2 Approximating Form Drag

Approximating form drag proves a more difficult challenge. Form drag is generated by the change in local pressures and flow around a body, resulting in mixing of streams of airflow and areas of high and low pressure. This varying pressure across the body generates a force which can be calculated by integrating the pressure difference across the body's surface area (What is Drag?, 2022). At a high level, differences in pressure are caused by deflection of fluid against the surface of a body, which increases with the angle of impact. Thus, a computationally simple approximation of form drag can be achieved by determining a metric associated with the angle between the surface of the body and the direction of flow. It is important to note that while we refer to a direction of flow, the water is relatively static, and the swimmer is moving through the water. This can, however, be visualized as a direction of flow opposite to the direction of movement of the body. We calculated such a metric and labeled it as the body's Pointiness Number, with a lower number representing higher pointiness, somewhat counter intuitively. However, pointiness has two forms, frontal and rear. Frontal pointiness represents the slope against the direction of flow for fluid colliding with the object, and rear pointiness represents the slope against the reverse direction of flow, representing how gradually fluid fills the void behind the object. The drag on spheres and hemispheres definitively shows this discretion *(tab. 1)*. For a sphere, front facing hemisphere, and rear facing hemisphere of the same maximum cross sectional area, their drag differs, as evidenced by their corresponding drag coefficients, a dimensionless quantity relating the frontal area of an object to the drag experienced by its form (Drag Coefficient, 2023). The sphere has the lowest drag coefficient, presumably due

Sphere	0.20
Frontal Hemisphere	0.42
Rear Hemisphere	1.17

Table 1: Drag coefficients of spheres and hemisphere("Table", 2023)

to a fairly low frontal and rear pointiness. The frontal and rear hemispheres, however, vary in drag coefficient, which led us to believe that frontal and rear pointiness contribute to overall drag in different amounts. Thus, we ascribed two coefficients to determine form drag, F for frontal pointiness, and B for rear pointiness. We then determined our equation for form drag to be:

$$F * FrontalPointiness + B * RearPointiness$$

To determine rear and frontal pointiness, we decided on a "slice and square" method. First, we take cross sectional slices of the body along the direction of flow *(fig. 5)*. The distance between slices is determined by a resolution constant, and drastically changes compute time and slightly changes accuracy. Very large resolution values create lost data, while resolution values too small result in irrational compute times. We chose a resolution value of 1 unit, which experimentally yielded the most accurate results for objects with known drag. Initially, a `frontalArea` variable is initialized as a cross section of size 0, which will come to represent the area water has already impacted. For each cross section, moving along the direction of flow, the cross sectional area not covered by the frontal area is squared and added to the `FrontalPointiness` value, and cross sectional area that was removed from the previous cross section is squared and added to `RearPointiness`. Squaring the additional value means that larger added areas hold higher significance, causing more drag. This step is what ensures that shallower angles create lower pointiness values, as they will result in less additional area each time.

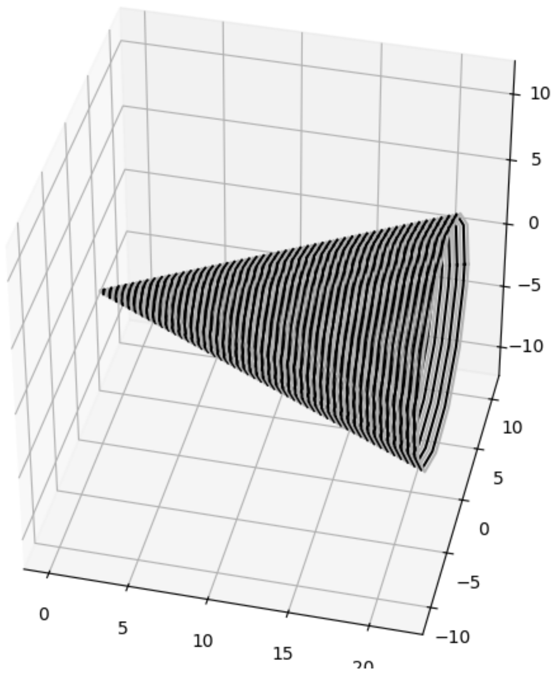


Figure 5: Cross sections taken from a cone.

9.2.3 Code

```python
#algorithm to calculate pointiness numbers based on cross
↪  sections. Height is the height of the object
while(i <= height):
    #determine the next cross section down from the direction of
    ↪  flow
    crossSection2 = prism.getCrossSection(i)
    #increment pointiness numbers appropriately
    pointinessNumberF += (getNewAreaFront(frontalArea,
    ↪  crossSection2)**power)
    pointinessNumberB += (getNewAreaBack(crossSection1,
    ↪  crossSection2)**power)
    #save the current cross section to memory to compare against
    ↪  next time
    crossSection1 = crossSection2
    #add new frontal area to total frontal area
    frontalArea =
    ↪  union_all([frontalArea,crossSection2]).simplify(0.1)
    #increment i to calculate for the next cross section
    i+=resolution
```

9.2.4 Finalizing

Once the algorithms and equations for both frictional and form drag were determined, we amalgamated a finalized drag approximation equation.

$$A * SurfaceArea + F * FrontalPointiness + B * RearPointiness$$

It then became necessary to determine the values of the coefficients and verify the drag model using known values.

9.3 Tuning and Verifying the Drag Model

To verify the drag model and determine the coefficient to each subset of the drag number, we determined a set of geometric shapes with known drag coefficients *(tab. 2)* and tested the shapes with our model to determine the correct ratios between the front pointiness factor, surface area factor, and rear pointiness factor. Cones were chosen as a baseline for front pointiness, the angle corre-

Sphere	0.20
Cone (20 degrees)	0.39
Frontal Hemisphere	0.42
Cone (40 degrees)	0.61
Cone (60 degrees)	0.83
Rear Hemisphere	1.17
Cone (90 degrees)	1.17

Table 2: Known drag coefficients used to tune the drag model ("Table", 2023)

sponding to one half of the vertex, the sphere was chosen to account for surface area drag, and the two bidirectional hemispheres account for the ratio between rear and front surface area drag. We then ran code to determine which rear pointiness coefficient and surface area coefficient values, given a front surface area coefficient of 1, yielded the same order as experimental data, and chose the median as our coefficients for the project *(fig. 6)*. Thus, complete with the coefficients determined to yield the best approximation of drag, we determined our equation to approximate drag for an object with reference area of 200 square units:

$$1 * FrontalPointiness + 0.5 * RearPointiness + 0.6 * TotalSurfaceArea$$

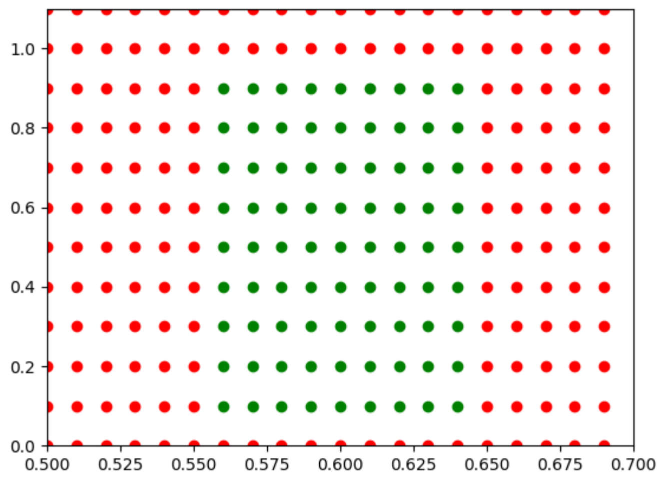


Figure 6: Coefficient values. Those yielding the same results as experimental data are shown in green. The X axis represents the `S` coefficient, and Y axis the `B`. The `F` coefficient is fixed at 1.

9.4 Applying the Drag Model to a Wireframe

With a `Swimmer` object successfully created based on the wireframe associated with an individual swimmer and a functional drag approximation model, the final step of determining the drag associated with an instance of the swimmer is extracting the necessary data from the `Swimmer` object to input into the drag approximation model. Having already determined the ratios between lengths of the swimmer's body by creating a wireframe, only one length must be manually measured. The left upper arm is thus measured from shoulder to elbow due to that measurement's ability to maximize comfort for the swimmer and minimize ambiguity in measurement points. With all lengths measured, widths are measured too for the limbs and torso of the swimmer. All wireframe lines are assumed to have circular cross sections for ease of computing, with this being relatively true for limbs, and accounted for by other factors for the torso, such as the shoulders covering the frontal area of the chest. Measuring widths is extremely important, as it individualizes the data to account for different body types and proportions, a distinct advantage of a computational system. Most widths are measured sideways along the X axis, from left to right relative to the swimmer, with the exception of two. Chest is measured from front to back along the Z axis due to the shoulders covering the front cross sectional X axis width, and Z axis variation generally occurring most prominently in the center of the torso. Shoulders are additionally measured along the Z axis due to their position on the human body perpendicular to the torso. These measurements are taken in addition to the wireframe to create a cylinder based exterior model of the swimmer's form *(fig #)*. The entire swimmer's wireframe and modeled form are then adjusted to a frontal area of 200 for a more accurate output from the drag approximation model.

To determine cross sections to calculate frontal and rear pointiness, each cylinder associated with a wireframe line is assigned to one of three categories: vertical along the Y axis, horizontal perpendicular to the Y axis, or angled along both axes. For example, the chest cylinder is generally vertical, the shoulder cylinders horizontal, and arm cylinders angled. When taking cross sections perpendicular to the Y axis to approximate drag, vertical cylinders will yield a circle cross section, horizontal cylinders a rectangle, and angled cylinders an ellipse *(fig #)*. Vertical cylinders are defined as those corresponding to a wireframe line parallel to the Y axis. Horizontal cylinders are those whose difference in the y values of the endpoints of their corresponding line do not exceed the cylinder's diameter. Angled cylinders fit neither category. The cross section of vertical and angled cylinders is determined based on the wireframe initially to optimize runtime, while the cross sections of horizontal cylinders are dependent on the y value of the cross section, and thus must be calculated by the drag approximation program. When the drag approximation program queries a cross section, the `Swimmer` object's `getCrossSection()` method first determines horizontal cylinders that intersect the queried cross section and return the corresponding rectangles, then returns the ellipsis and circles generated by the intersection with vertical and angled cylinders, adjusted for the location in the cross section they would appear. The method then returns the collection of shapes making up the cross section as a `Multishape` in the Shapely polygon handling library *(fig. 7)*.

9.4.1 Code

```
#method to get cross section of swimmer object
def getCrossSection(self, height):
        #initialize all shapes of crosssection
        shapes = []
        #adjust height to shape height
        height = self.minheight + height
```

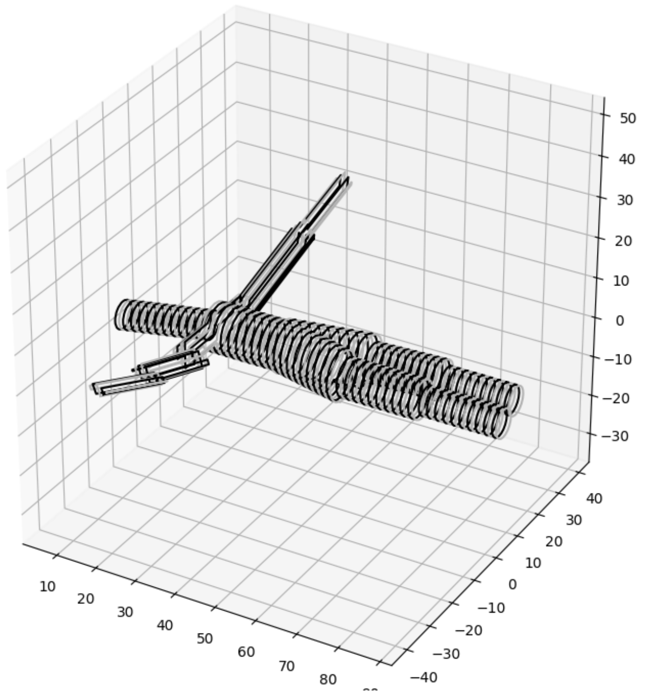


Figure 7: Cross sections taken from a wireframe of an improperly streamlined swimmer.

```
        #iterate through lines of wireframe
        for i in range(len(POSE_PAIRS)):
            #get shape of cross section. The midpoint for
            ↪  horizontal cylinders, and the cross sectional
            ↪  shape otherwise
            shape = self.xsections[i]
            #get points of the line of wireframe
            pair = POSE_PAIRS[i]
            pointA = self.wireframe[pair[0]]
            pointB = self.wireframe[pair[1]]
            #get the diameter of the cylinder
            diameter = diameters[i]
            #if horizontal cylinder
            if (type(shape) is list):
                #get distance of slice from midpoint
                #if less than radius add to shapes as rectangle
                xdistance = height - shape[0]
                ydistance = np.sqrt((diameter/2)*(diameter/2) -
                ↪  xdistance*xdistance)
                if(ydistance < diameter/2):
                    getPointsinRectangle(pointA, pointB)
                    shapes.append(Polygon(points))
            #if greater than radius add to shapes as ellipse at correct point
            else:
            #determine top and bottom point
                if(pointA[0] > pointB[0]):
                    bottom = pointB
                    top = pointA
                else:
                    top=pointB
                    bottom=pointA
                #insert ellipse if the cross section crosses the line
                if(bottom[0] <= height <= top[0]):
                    #determine at what fraction along the line the cross section lies
                    fraction = (height - bottom[0])/(top[0] - bottom[0])
                    #adjust x and z values of ellipse accordingly
                    x = bottom[1]+(fraction*(top[1]-bottom[1]))
                    z = bottom[2]+(fraction*(top[2]-bottom[2]))
                    #move the ellipse to the determined center and append
                    shapetransformed = transform(shape, lambda a: a + [x, z])
                    shapes.append(shapetransformed)
        #return shapes
        return union_all(shapes)
```

To optimize computational runtime, all swimmer objects are assumed to have the same surface area, as lost surface area to adjacent touching limbs is minimal, and thus the arbitrary surface area value of 1 is returned.

10 Determining Optimal Position

10.1 Creating a Base Wireframe

Initially, a base wireframe is constructed using the lengths between points on the wireframe. The entire body is positioned flat on the XY plane, and the distance between points is used to construct an initial starting point for the wireframe with legs directly backwards along the Y axis and arms directly forwards *(fig #) (fig. 8)*. The purpose of this is twofold. First, constructing a base wireframe ensures that the optimization algorithm does not encounter any "valleys" of optimization. For example, despite pointed and slanted arms above the head being a more hydrodynamic position, to achieve that position from arms at the side, a swimmer would need to pass their arms through a position with each arm extended to the side, less hydrodynamic than both the start and end positions. By starting with a base wireframe, we allow the use of an optimization algorithm that makes tweaks towards a final product, as opposed to testing every possible configuration of points, an incredibly computationally intensive task. Second, a tweak-based algorithm is incredibly susceptible to initial errors, which are not only unavoidable but exactly what we are seeking to address with our model. For example, a slightly upwards rotated left leg may influence a left arm to move upwards to address this error, especially if the left arm is tweaked before the leg. Consequently, the right arm may move downward to address left arm movement. Such an effect can quickly create undesired results, and a base wireframe addresses this. Within the `Swimmer` class, we define a `constructBaseWireframe()` method that returns a base wireframe based on the lengths taken from the original wireframe. This is done recursively by starting at the chest point, calculating distances, adjusting points accordingly, and performing a similar operation for each adjacent point, and each adjacent point after that.

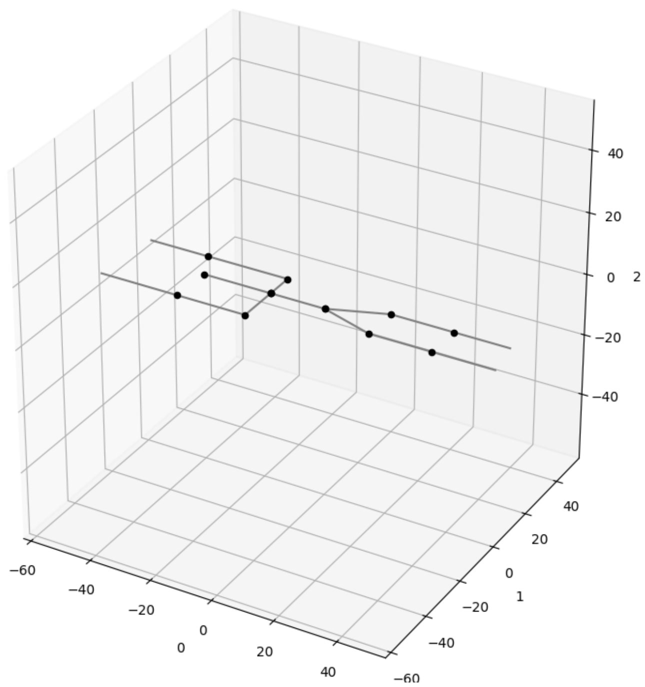


Figure 8: An example base wireframe.

10.2 Tweaking the Wireframe

Wireframe tweaking is accomplished using quaternions, mathematical operators for rotating vectors (Hughes, 2017). In general, they are a set of operations and specific rules based on the multiplication of hyper-complex numbers analogous to the i value that yield rotations of vectors. We use the PyQuaternion library to perform rotations of points due to the simplicity and ease of computation of quaternions. Within our `Swimmer` class, we define a `rotateWireframe()` method that takes the `Swimmer` object and rotates one point around another, subsequently rotating all connected points to maintain distance and allow rotation of entire limbs or sections. We wrote a constant `DEPENDANTS` tree that is used to determine the points that must be rotated with a point, and access those points recursively through a function.

10.2.1 Code

```
#method to rotate a section of wireframe. Basepoint is the point
↪  to rotate around, point2rotate the point to rotate, and y,x,
↪  and z the degrees of rotation
def rotateWireframe(self, basepoint, point2rotate, y,x,z):
        #retreive wireframe
        wireframe = self.wireframe.copy()
        #define axis for the quaternion
        yaxis = [1,0,0]
        xaxis = [0,1,0]
        zaxis = [0,0,1]
        #multiplication of quaternions results in a single
        ↪  quaternion with the same result as performing each
        ↪  rotation seperately
        quaternion = q.Quaternion(axis=yaxis,angle=np.deg2rad(y))
        ↪  *q.Quaternion(axis=xaxis,angle=np.deg2rad(x)) *
        ↪  q.Quaternion(axis=zaxis,angle=np.deg2rad(z))
        #determine the points in the subsection that must be
        ↪  rotated along with point2rotate
        subsection = getSubsection(point2rotate)
        pointsInSubsection = getPointsInSubsection(subsection)
        #rotate both shoulder points together, and likewise with
        ↪  hips
        if(point2rotate == 2):

            ↪  pointsInSubsection.extend(getPointsInSubsection(getSubsection(5)))
        elif(point2rotate == 8):

            ↪  pointsInSubsection.extend(getPointsInSubsection(getSubsection(11)))
        #rotate all points, first zeroing them around the
        ↪  basepoint, and then returning them to their original
        ↪  coordinates
        for point in pointsInSubsection:
            zeroedPoint = np.subtract(wireframe[point],
            ↪  wireframe[basepoint])
            rotatedZeroedPoint = quaternion.rotate(zeroedPoint)
            rotatedPoint = np.add(rotatedZeroedPoint,
            ↪  wireframe[basepoint])
            wireframe[point] = rotatedPoint.tolist()
        return wireframe
```

10.3 Optimization Function

To optimize a wireframe generated through computer vision into the hydrodynamically optimal wireframe, we include recursive iterations of the previously illustrated functions. Initially, we use the `constructBaseWireframe()` method and set the wireframe of the `Swimmer` object to the method's return value. Next, the optimization loop begins and is repeated until no change in wireframe is detected.

10.3.1 The Optimization Loop

One iteration of the optimization loop initially begins with the rotation of the sections connected to the chest point of the swimmer. First, each point connected directly to the chest point is rotated five degrees in either direction around the X axis and the magnitude of the calculated approximate drag is collected. The drag values are compared against each other and the initial value, and the most optimal result, yielding the lowest drag, is used. This process is repeated for rotation around the Z axis. Only two axis are chosen because any rotation of a line from an endpoint around the Y axis can be replicated by two turns around the X and Z axes. For each point rotated, the same process is recursively applied to each of their connected points, and so forth. In effect, the upper body and lower body are each rotated to their optimal position by five degrees, followed by the shoulders and hips, which are followed by the arms, legs, and head. Five degrees is chosen as the minimum value visible and correctable by a human swimmer, and is maximized to improve compute times. The optimization loop is then repeated.

10.3.2 Error Accounting

A few amends to the optimization function were made to account for common errors.

- The chest and shoulders are not rotated on the first pass through of the optimization loop. To create slant in the arms, a quality which is desirable through arm rotation, the algorithm would rotate the upper body an arbitrary direction on the first pass through. Allowing the arms to rotate before the chest and shoulders ensures that arm slant is created through arm rotation, and preserves the ability to use a tweak-based optimization algorithm.

- The head is not optimized until after the rest of the body. As the arms rotated in towards a middle ground, the head was attempting to fill the space of the arms not yet rotated to cover the head, and was tilting drastically to the side.

10.3.3 Code

```
#function to optimize the wireframe
def minimizeDragNumber(swimmer : SC.Swimmer):
    #set the swimmer's wireframe to its base wireframe
    swimmer.setWireframe(swimmer.constructBaseWireframe())
    #initialize old wireframe and new wireframe to compare
    ↪  between iterations
    wireframeold = []
    wireframenew = swimmer.getWireframe()
    #conduct an initial improvement of the wireframe, excluding
    ↪  chest and shoulders. Each call of the improveWireframe
    ↪  method decrements the second argument by one. Due to the
    ↪  recursive nature of the function, the shoulders and chest
    ↪  are not optimized the first time through due to the
    ↪  positive second argument value
    improveWireframe(swimmer,2)
    #repeat iterations until no change is detected
    while(wireframenew != wireframeold):
        #print statement to see progress of optimization
        print(dN.getDragNumber(swimmer))
        wireframeold = wireframenew.copy()
        #the improve wireframe method tweaks each joint 5 degrees
        ↪  in the most optimal direction
        improveWireframe(swimmer,0)
        wireframenew = swimmer.getWireframe()
    return wireframenew
```

11 User Friendly Information Display

11.1 Wireframe Visualization

Lastly, we implemented a visualizer to relay the information to the user in a visually intelligible form. Both the three-dimensional wireframe taken from the swimmer's position and the hydrodynamically optimized wireframe are graphed, including the constituent points and the lines connecting them. The original wireframe is graphed in red, and the optimized wireframe is graphed in green *(fig. 9)*.

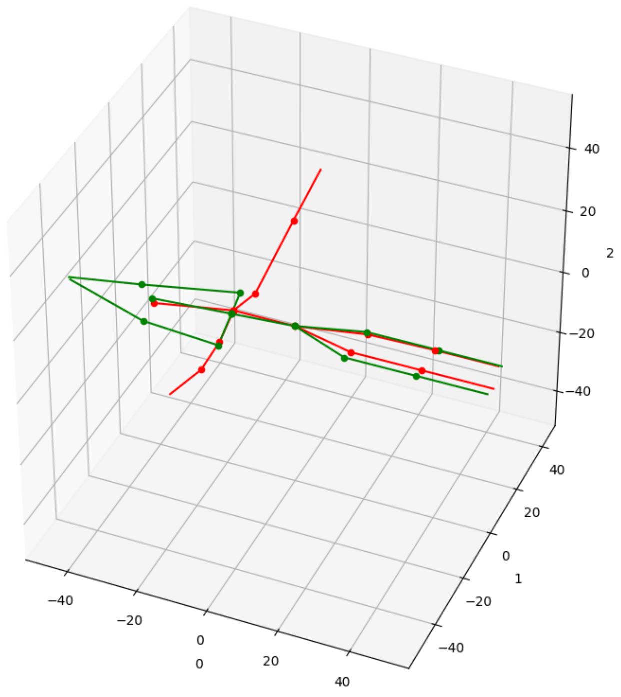


Figure 9: An example output.

Verifying the Efficacy of our Method

12 Criteria for an Optimal Streamline

As previously outlined, an optimal streamline consists of arms above head, hands together and legs straight behind, the body forming a shape reminiscent of a torpedo or dolphin, a shape backed up by both engineering concepts and a multitude of years of evolution. Thus, we define criteria for an effective streamline accordingly. To determine the efficacy of our model, we compare results to the following set of criteria.

- The arms of the swimmer are outstretched above the head, and angled inwards towards the body's center line.

- The hands of the swimmer are connected or near connected, and the ends of the forearms are overlapping.

- The legs of the swimmer are stretched behind the swimmer, minimizing total frontal area within possible constraints of the human body.
- Frontal area appears to be minimized in all ways possible within the constraints of human form.

13 Verification of the Model

To verify the model, we provided 3 sets of images to the model and analyzed the outputs. To create a variety of test cases, we used two streamlines most commonly created due to errors, and one that was attempted to be correct *(fig. 10)*. Both optimized wireframe (a) and (b) fit the criteria above. Optimized

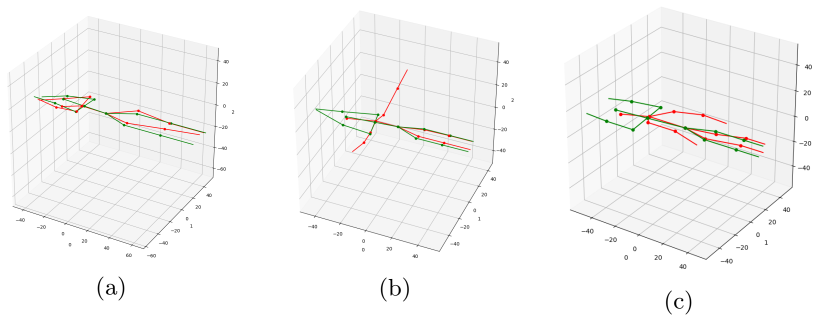


Figure 10: (a) A test case for a correct streamline (b) A test case for an incorrect streamline (c) A test case for an incorrect streamline with issues that will be discussed in the next section

wireframe (c), however, fails many of the criteria, but yet surprisingly still represents a successful algorithm, albeit highlighting errors in data collection.

13.0.1 Wireframe Optimization Inaccuracies

It is visible that optimized wireframe (c) does not fit the criteria of hands connected or near connected, and this issue mainly arises from data collection accuracy issues, further discussed in Section 14. The arms in this figure are collected as shorter than the other figures, while the shoulders are collected as wider. The main difference however, is that the shoulders were relaxed down, as opposed to hunched upwards as in wireframe (a) and (b). The image recognition software is unable to recognized hunched upwards shoulders as a change in the angle between shoulder center and the end of each shoulder, and just represents these intricacies as wider or skinnier shoulders, incorrectly ascribed intrinsic to the person. Therefore, the model neglects the ability to hunch shoulders upwards as a consequence of data collection limitations. The drag model, thus, determines that due to the short nature of the arms without shoulder movement coupled with wide shoulders, the drag minimization derived from moving arms towards a center line is actually sub optimal. Shoulders have a natural slope to them, and moving the arms towards the center line would increase the drag on the arms by causing a flatter surface. The drag model has determined that by covering some percentage of the shoulders by a slope in the arms while leaving some shoulder exposed to the flow, drag is actually optimized *(fig. 11)*. Improvements to data collection as well as improvements to defining the shape and limitations of the human body would improve such an issue. Additionally, this highlights an interesting specific use case for the model.

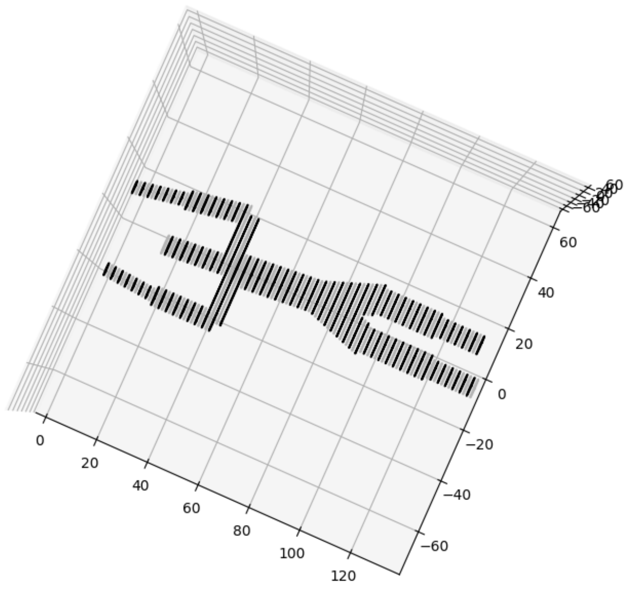


Figure 11: Cross sections of optimized wireframe (c). Note the slope of the shoulders and slope of the arms.

13.0.2 Ability of the Model to Account for Differences and Disabilities

Take a swimmer who had mobility issues due to a prior injury or surgery. Additionally, take a swimmer who had birth differences leading to decreased mobility and different body proportions. Scenarios such as these could easily be accounted for by our model. Wireframe (c) inadvertently could represent a person with an injured latissimus dorsi, the main muscle along a person's back, or rotatory cuff, the area of tendons around the shoulder, who is unable to lift their shoulders. Aforementioned fixes to the model could increase its ability to accurately assist individuals without movement issues, but the reintroduction of restrictions could even further individualize the model for individuals with limitations. Wireframe (c) demonstrates that drastically different results are optimal for individuals with differences. Shoulder rotation limitations could be introduced to account for injuries. Differences in body proportions will be picked up by the computer vision algorithm. Individuals who could have previously struggled with coaching individualized to their needs would receive personalized results computationally by CHASM.

Finalizing Thoughts

14 Data Collection Improvement Needs

As previously stated, the CHASM model is incredibly susceptible to irregularities in position and wireframe. This is a desired quality, as such irregularities influence the position of an optimal hydrodynamic wireframe, and enforce the need for individualized computation as opposed to comparing to a universal baseline. This sensitivity, however, leaves the model highly sensitive to data collection inaccuracies. In our physical LVU, imperfections in measurement could extrapolate to noticeable inaccuracies. A better production method for the LVU than what was accessible to us using household power tools would improve data collection and address many of these inaccuracies, which included egregious differentiation between widths of right and left shoulders, and inaccuracies in point location due to improperly angled cameras.

15 Further Steps and Project Potential

With the production of a more reliable LVU, data collection accuracy would drastically improve. Further steps in improving computational accuracy could be taken by upgrading the wireframe model to a surface based model of the swimmer's form. This would allow the computation of more accurate drag, as well as determining forces added to the water, opening the doors to aspects of the sport of swimming other than the streamline. Freestyle optimization would be a logical, yet difficult, next step, and would likely require the use of more complicated hardware and software in order to collect accurate data for a body in relative motion to itself and account for forces added against the water. Additionally, introduction of restrictions to movement could specialize the model towards individuals with injuries or disabilities. With improvements, the steps and algorithms outlined above could provide coaching augmentation and replacement for almost all aspects of swimming, increasing accessibility and productivity of the sport.

16 Conclusion

As the project currently stands, CHASM provides visual feedback on the streamline portion of swimming. It can improve the ability of coaches to give meaningful feedback to their swimmers, and increase accessibility to a sport that involves paywalls due to coaching access. CHASM uses computer vision, mathematical coordinate operations, a drag approximation model, optimization functions, and visual output to accurately and effectively communicate feedback on a swimmer's streamline position. With improvement, CHASM could be expanded to provide feedback on all aspects of swimming, providing an invaluable coaching aid to augment the needs of competitive and recreational lap swimmers of all levels, and completely replace a coach for those who do not have the means to have one. Additionally, access to personalized coaching for individuals with disabilities or injuries that effect their body or abilities could exponentially increase. Adoption of CHASM, as it stands or as it could be, by swimming pools, coaching teams, individuals, and schools could prove an asset to creating greater depth of field and accessibility in the sport, all the while improving the form of preexisting participants. CHASM is a means to accessibility and inclusion for underprivileged, injured, and disabled individuals, effective coaching for swimmers of all levels, and speed maximization for the sport of competitive lap swimming.

17 Acknowledgements

Invaluable in our coding, building and development process were our wonderful sponsors Ms. Masoni and Ms. Comstock. Without their assistance, not only would we not have a designated time during the school day to work on our project, but we would not have the guidance and experience they bring to the table. Additionally, we would like to acknowledge Marlow Lichty, a prior team member from earlier in the year who elected to leave our team to invest his time elsewhere, but nonetheless contributed important work in the implementation of the image processing software.

18 Works Cited

References

[1] 360swim.com, C. @. (n.d.). Streamline explained (how do drag forces influence my body in swimming?). 360swim. https://360swim.com/blog/streamline-explained-how-forces-influence-swimming#google_vignette

[2] Admin, F. S. (2021, December 13). Why streamline position is so important. Precision Swim Training. https://www.precisionswimtraining.com/blogs/kicking-with-kaitlin/why-streamline-is-so-important

[3] Andriluka, M., Pishchulin, L., Gehler, P., & Bernt, S. (2014). 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Dougherty, A. (2017, June 14). A cost benefit analysis of a lifelong swimming career. Swimming World News. https://www.swimmingworldmagazine.com/news/a-cost-benefit-analysis-of-a-lifelong-swimming-career/

[5] Drag coefficient. (2023). Retrieved from https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/drag-coefficient-2/

[6] Gupta, V. (2018, May 29). Deep Learning based Human Pose Estimation using OpenCV. LearnOpenCV. Retrieved April 2, 2024, from https://learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/

[7] Here is how swimmers of all levels train (according to coaches). A3 Performance. (n.d.). https://www.a3performance.com/blogs/a3-performance/here-is-how-swimmers-of-all-levels-train-according-to-coaches

[8] Holmes, T. (2022). How to have perfect streamline in swimming. Retrieved from https://blog.myswimpro.com/2022/02/08/how-to-have-perfect-streamline-in-swimming/

[9] Hughes, M. (2017). Don't get lost in Deep space: Understanding quaternions - technical articles. Retrieved from https://www.allaboutcircuits.com/technical-articles/dont-get-lost-in-deep-space-understanding-quaternions/

[10] Koury, J. M. (2024, March 25). Individual coaching for competitive swimmers - swim technique coaching Lehigh Valley - swim articles. Swim Technique Coaching Lehigh Valley. https://www.swimtechniquecoaching.com/swim-articles/2023110private-coaching-for-competitive-swimming-athletes

[11] Laminar and turbulent flow. (1992). In (Ed.), DOE Fundamentals Handbook: Thermodynamics, Heat Transfer, and Fluid Flow. U.S. Department of Energy. https://engineeringlibrary.org/reference/laminar-and-turbulent-fluid-flow-doe-handbook

[12] OpenStax College. (n.d.). Motion of an Object in a Viscous Fluid. Lumen Learning. Retrieved April 2, 2024, from https://courses.lumenlearning.com/suny-physics/chapter/12-6-motion-of-an-object-in-a-viscous-fluid/

[13] Poirier-Leroy, O. (2023, December 19). How to streamline in swimming like a pro (swim faster and glide farther). SwimSwam. https://swimswam.com/streamline-in-swimming/

[14] Table 10.5 presents the drag coefficient, CD for numerous... (1 answer). (2022). Retrieved from https://www.transtutors.com/questions/10-23bg-table-10-5-presents-the-drag-coefficient-cd-for-numerous-geometric-shapes-at-6923663.html

[15] The importance of a good streamline. SAN. (n.d.). https://www.swimacademynetwork.com.au/resources/the-importance-of-a-good-streamline

[16] Ungerechts, Bodo. (1982). The Validity of the Reynolds Number for Swimming Bodies Which Change Form Periodically.

[17] What is drag? (2022). Retrieved from https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/what-is-drag/

[18] Wise, J. (2022). Role of technology in swimming: The good and bad. Retrieved from https://www.swimmingworldmagazine.com/news/role-of-technology-in-swimming-the-good-and-bad/

[19] Wynn, K. (n.d.). Welcome. Retrieved from https://kieranwynn.github.io/pyquaternion/

New Mexico

SUPERCOMPUTING CHALLENGE

ROBOTIC GUNSHOT DETECTION SYSTEM: SCOOBY

Capital High School

Team: SCOOBY (CHS 1)

Team members:

Raul Alvarado Villalobos

Craig Andree Abajo

Jesse Burch

Haize Christiansen

Jordan Lam

Britny Marquez

Edward Scott

Teacher Sponsor:

Barbara Teterycz

Mentor:

David Ritter

April, 9 2024

Table of Content

Abstract/Executive Summary

Our country has been challenged with an ongoing and escalating issue of mass shootings in schools across the United States. According to the Center for Homeland Defense and Security, there have been 2069 school shootings in our country between 1970 and June 2022 with 684 killed. The list of schools is long: Virginia Tech, Sandy Hook Elementary, Robb Elementary, University of Texas-Austin, Parkland High, Columbine High, Texas Santa Fe High, Umpqua Community College, Red Lake Senior High, and hundreds of other schools. The deadliest year was in 2018, where 51 people were killed. In response to this critical need our team has been developing a robotic gunshot detection system, called SCOOBY, that can recognize the sound of a gunshot, precisely locate a shooter on the first shot, and set off appropriate alarms and distractions.

The project first started off with an Arduino Nano, but over time, the team learned that the Nano just did not have enough memory to complete the calculations needed for precise timing and location. The project was moved onto a Raspberry Pi microcontroller, which contained more memory and much higher processing speed. The peripherals connected to the microcontroller are: 4 directional microphones, 1 waveform microphone, an LCD display, SD drive, real time clock, mic amplifiers, comparators and a printed circuit board. Scooby's software was written using C++ and Python and has been expanded to test new ideas for the system. After the project was built and programmed, it was tested in various scenarios/experiments, and the data was collected and analyzed. As a safer and more reasonable way of testing the project, the team decided to use popped balloons at higher pressure thresholds than normal, as their sound waves are very similar to a gunshot.

After testing SCOOBY, it was able to correctly point towards the balloon as well as discern the difference between a balloon and an everyday sound. The key features (characteristics of the waveform) identified during data exploration and analysis include: leading edge time, peak, peak duration, and the zero crossing time. By installing a system of SCOOBY units across a school, and having them communicate to a main hub via wires, schools can determine if a loud sound was a gunshot and where it was located. This will prevent the future deaths of students and staff.

Research and Background

School shootings in our country have become an everyday occurrence. In 2022, there were 240 non-active and active shooter events. This was the deadliest year here according to the Center of Homeland Defense and Security School Safety Compendium. Here in New Mexico, there have been 13 school shootings which led to 6 deaths and 6 injuries. Some of the deadliest school shooting events in the US include Virginia Tech (33 killed in 2007), Sandy Hook Elementary (28 killed in 2012), Robb Elementary (21 killed in 2022), University of Texas-Austin (18 killed in 1966), Parkland High (17 killed in 2018), Columbine High (15 killed in 1999), Texas Santa Fe High (10 killed in 2018), Umpqua Community College (10 killed in 2015), Red Lake Senior High (10 killed in 2005). The statistics of the number of students and staff killed during these events are grim. It's important that schools have security systems that can save people's lives in such events.

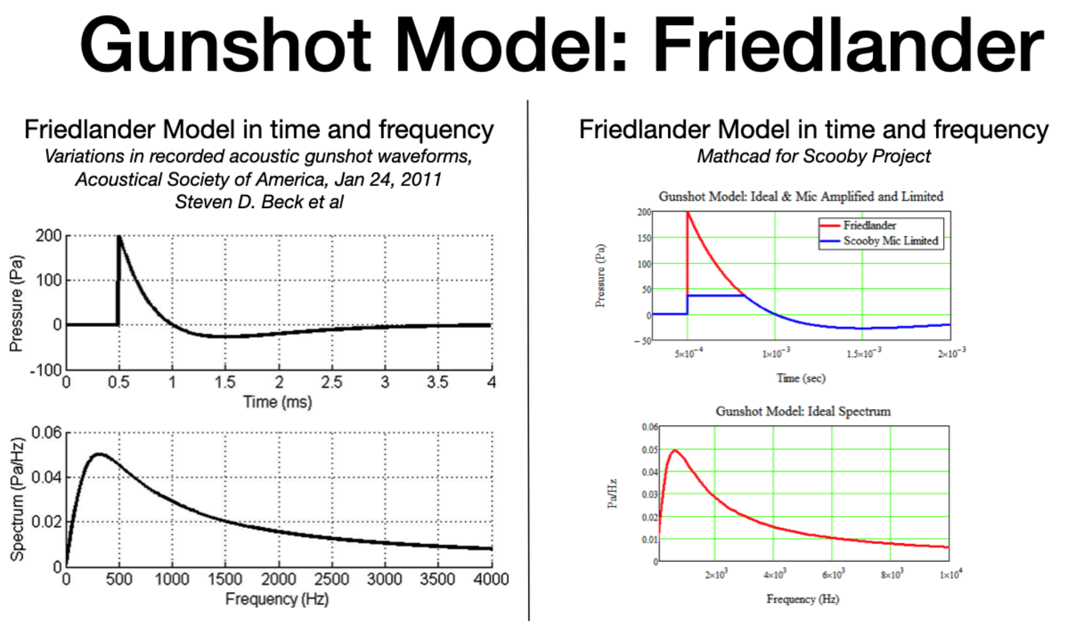
The inspiration for this project came from a gunshot detection system which was implemented in Chicago in August 2018 by a company called Shotspotter. This system was implemented over 117 square miles with 15-20 sensors placed per square mile. When their sensors detected a loud sound, it determined whether the sound was a gunshot. After it verified the sound as a gunshot, the system alerted local authorities and informed where the gunshot was located. This system contained many flaws which led other cities to drop contracts with Shotspotter. Safety officials questioned the efficacy of the system, since only 1 out 9 alerts included a gun. This system was costly, the city of Chicago signed a 3 year, $33 million contract for Shotspotter to implement this system. Also, a judge in Manchester, New York dismissed evidence provided by the Shotspotter system in weapon conviction since it "was not reliable scientific proof". Shotspotter is also selling its system to school districts and individual schools, but there are many privacy concerns. People are afraid it could pick up people's conversations, leading it to become a spying device. SCOOBY takes all these factors into consideration. For example, it cannot record a human voice, it is bandwidth (1kHz) and time limited (2msec).

The Acoustical Society of America used the Friedlander Model to study gunshot characteristics in their paper "Variations in recorded acoustic gunshot waveforms generated by small firearms". This model produces the ideal gunshot pressure waveform. The Friedlander model is given by this equation for the pressure wave after 'tstart':

$$p(t) := \left[P0 + Ps \cdot \left(1 - \frac{t - tstart}{T0}\right) \cdot e^{-b \cdot \frac{t - tstart}{T0}} \right]$$

Where:

- p(t) is the pressure at time t,
- P0 is atmospheric pressure, set to zero here to reflect gauge pressure (re atmosphere),
- Ps is the peak pressure, set to 200 Pa (pascals) to reflect a sound level peak of 140dB (decibels) SPL (sound pressure level)*,
- T0 is the positive duration of the pulse, set to 500 μsec,
- e is Euler's number ≅ 2.718281828459045,
- b is the exponential decay constant, set to 1 (typical),
- tstart is the time that positive pressure (time effectively zero) starts.

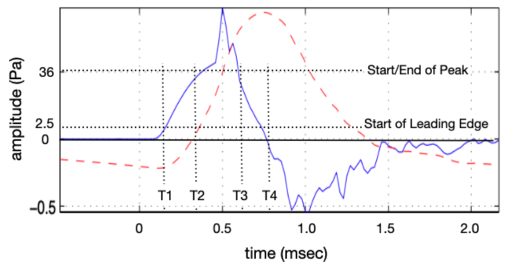


The waveform of a gunshot is characterized by the sudden rise of pressure at start then followed by a peak and exponential decay. Major features are the length in time of the gunshot (~2msec), and the frequency content of the waveform (substantially below 1000Hz). The

Friedlander model was coded into Mathcad so that the limitations of SCOOBY’s microphone and signal processing can be included. The blue curve shows the waveform that arrives at SCOOBY’s ADC (Analog to Digital Converter). Although the absolute peak is lost in this implementation, its nature is still reflected in the duration of the peak which is a key feature used in SCOOBY’s code.

However, due to safety protocol, gunshots have been substituted for the balloon pops (since their acoustic properties are so similar). Both sounds have a SPL (sound pressure level) peak of about 140dB (decibel) and a peak duration of about 500 µsec.

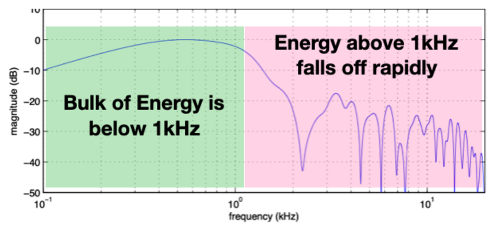
Balloon Pop Waveforms and Spectrum



Waveform Properties

1. Leading Edge: is less steep than a gunshot but still in reasonable range for Scooby. We don’t use the slope of leading edge, but just look for the 50% point to identify the start of the event relative to the trigger point from other microphones.
2. Peak: length of the peak from balloon pop is similar to gunshot and falls in the range around 350usec. This a key feature of both balloons and guns and they are very similar.
3. Zero Crossing: for a valid waveform it must cross zero after the peak. This is also a key distinguishing feature of both balloon pops and gunshots.

Note: red line is the integral of the waveform (used in source paper) which we don’t use here.



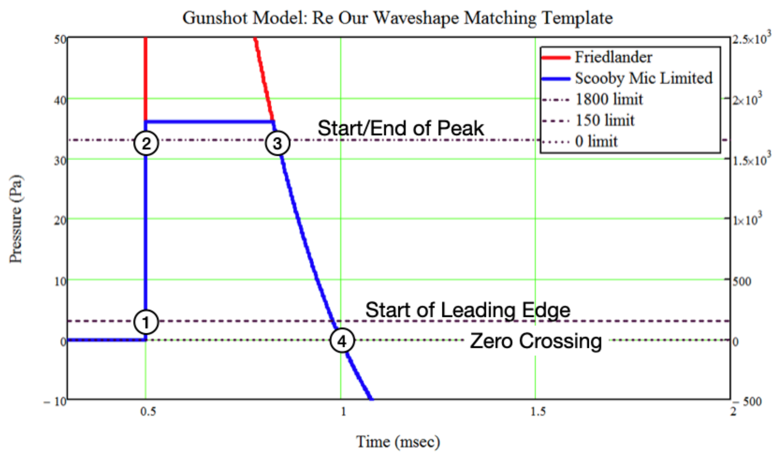
Spectrum Properties

1. Most of the energy is below 1kHz, so we can apply filters to ignore higher frequencies. This is very similar to gunshots.
2. Energy above 1kHz falls off rapidly. We can ignore this content.
3. Our trigger microphones have filters included to remove frequencies above 1kHz and emphasize those below. This helps prevent false triggering on random loud sounds.

Balloon pops (balloon inflated consistently to bursting pressure using an automated inflator fixture) are used as surrogates for gunshots because they have similar characteristics. A balloon pop has a rapid leading edge waveform, followed by a peak and a zero crossing with timing similar to a gunshot. Studies show that a balloon inflated until it bursts has the same acoustic energy as a gunshot.

The figure below demonstrates the details of the relevant part of the waveform (less than 2 msec). In this region only, the direct sound from the source reaches the microphone. There are no reflections to distort the waveform. In this plot the important amplitudes of SCOOBY’s discriminator are shown.

Gunshot Model: Detail and Matching



Key Features

1. Leading Edge: characterized by start of leading edge (point 1) and start of peak (point 2).
2. Peak: characterized by start (point 2) and end (point 3) of peak.
3. Zero Crossing: characterized by waveform crossing zero (point 4).

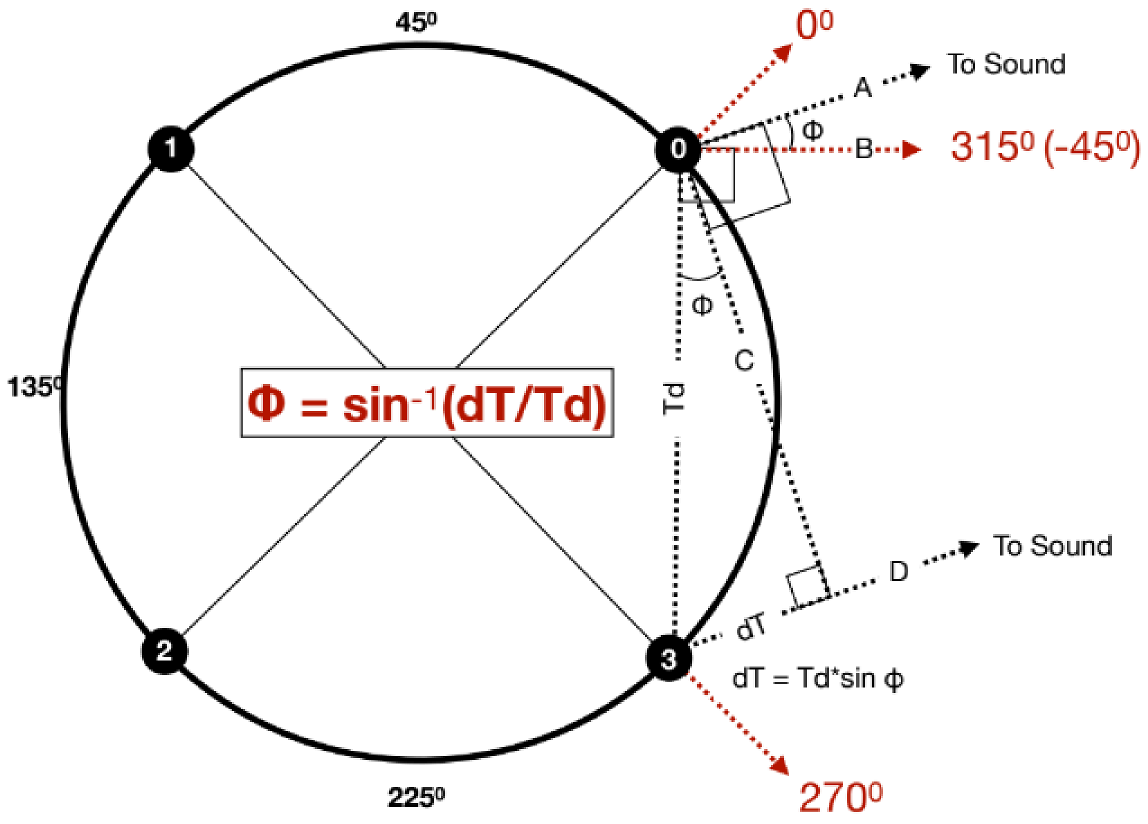
Main Goal

The main goal of this project is to develop a robotic gunshot detection system called SCOOBY that can recognize the sound of a gunshot, precisely locate a shooter on its first shot, set off appropriate alarms and distractions. The plan is to make multiple units of SCOOBY and make it as cheap and accessible as possible, so schools can easily implement it into their respective security systems.

Selected Approach

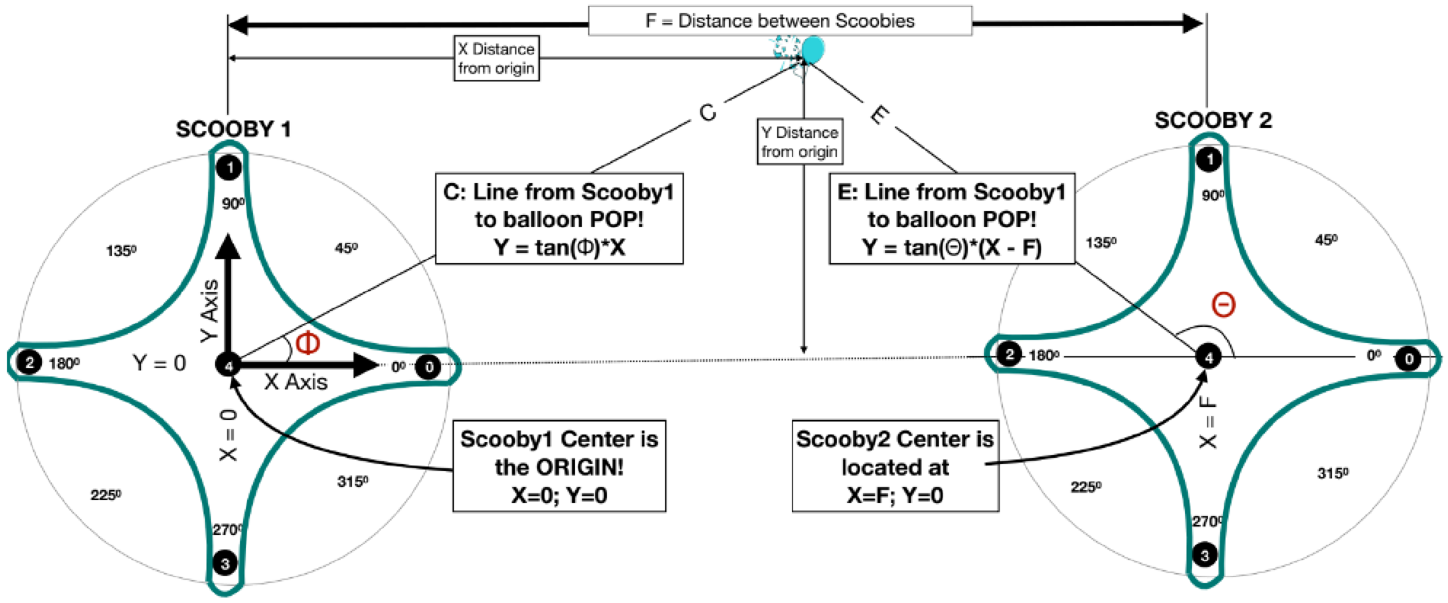
Angle Calculation

SCOOBY calculates the angle of a gunshot relative to its microphones using geometry and trigonometric functions. All of the precise timing to accomplish this is included in the SCOOBY unit. It does this with the use of the first two microphones on the unit being triggered. In the diagram (below) lengths are labeled as times by applying the speed of sound. So dT (difference time), while it is a time, is proportional to the extra distance a sound has to travel between microphones (in this case Mic3 to Mic0). The distance between mics at Td (timed distance) is calibrated as a distance which again is a time proportional to distance. C is simply the distance between microphones calibrated in delay time between them. This defines a triangle [Td, dT, C] and an angle Φ. Given these assumptions, the sin of Φ is simply dT/Td, and Φ is computed as the inverse sin of dT/Td, shown in the diagram below. This method has the advantage of keeping all critical dimensions and timing inside a single SCOOBY. SCOOBY reports events and angles, therefore multiple SCOOBIES can triangulate without requiring critical synchronization.



Triangulation

SCOOBY would be connected into a main hub for multiple units to send data at once. This data will then be used to triangulate a gunshot within a room or other space. For basic triangulation two units are needed. Each unit reports the angle to the detected sound. These angles (along with the position and separation of the units) will then be used to calculate the coordinates of the detected sound.



F = Distance between Scoobies
X Distance from origin
Y Distance from origin
C
E
SCOOBY 1
SCOOBY 2
C: Line from Scooby1 to balloon POP! Y = tan(Φ)*X
E: Line from Scooby1 to balloon POP! Y = tan(Θ)*(X - F)
Φ
Θ
Y Axis
X Axis
Y = 0
X = 0
X = F
Scooby1 Center is the ORIGIN! X=0; Y=0
Scooby2 Center is located at X=F; Y=0
0º
45º
90º
135º
180º
225º
270º
315º
1
2
3
4
0

Two lines are drawn from each unit to the location of the detected sound: C for the first unit and E for the second. They intersect at the location of the detected sound. The center of the first unit defines the origin of the coordinate plane, (0,0). The second unit is at a known distance of F, with its center at (F,0). The angle of the first unit to the detected sound is phi and the angle from the second unit is theta. The equation of each line is y = mx + b, which can be rewritten as y = m (X-X0), where X0 is the X intercept of the line. The slope of line C is m = tan(Φ) and for line E is m = tan(Θ). The equation for line C is, y = tan(Φ) * X, while for line E it is, y = tan(Θ) * (X - F). These two equations are then used to calculate the X coordinate.

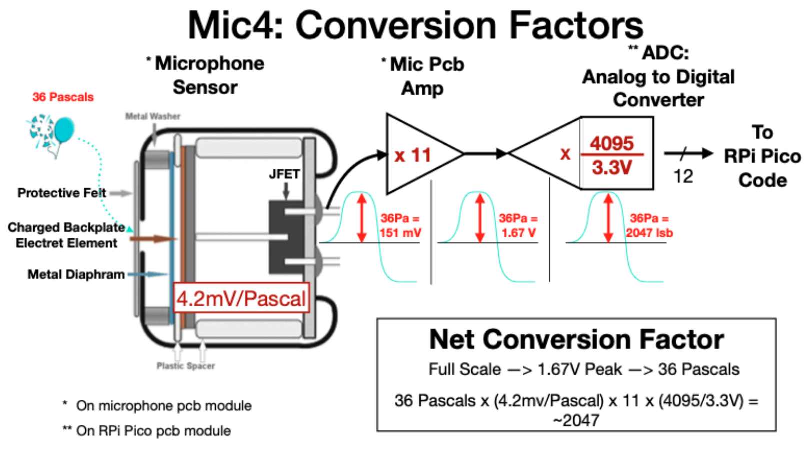
1. Slope of C is tan(Φ), Slope of E is tan(Θ)
2. Equation of C: Y = [tan(Φ) * X]
3. Equation of E: Y = [tan(Θ) * (X - F)]
4. [tan(Φ) * X] = [tan(Θ) * (X - F)]
5. [tan(Φ) * X] = [tan(Θ) * X] - [tan(Θ) * F]
6. [tan(Θ) * X] -]tan(Φ) * X] = [tan(Θ) * F]
7. [tan(Θ) - tan(Φ)] * X = [tan(Θ) * F]
8. X = tan(Θ) * F / (tan(Θ) - tan(Φ))
9. Y = tan(Φ) * X ... from (2)

The formula to calculate the X coordinate will always be x = tan(Θ) * F / (tan (Θ) - tan(Φ). There are some instances when triangulation could fail. If tan(Φ) = tan(Θ) then the angles to the detected sound are equal and the sound is too far away for a meaningful result. By the time the wave hits the microphones, the curvature of the wavefront is gone and the sound hits both microphones at the same time. This is a parallel error: the sound is so distant that C and E are effectively parallel. It also fails anywhere along the X axis where both tan(Φ) = 0 and tan (Θ) = 0.The equation for X results in a 0/0 indeterminate form..

In the remaining cases there are two places where the calculation becomes less reliable. Small errors in angle can make huge differences. To avoid these cases there are two ways to calculate Y from the X solution. The first one is when |tan(Θ)| < |tan(Φ)| or when the detected sound is closer to the first unit. In this case it is advantageous to calculate the Y coordinate using: Y = tan(Θ) * (X - F). While for the second case, when |tan(Θ)| > |tan(Φ)| or when the detected sound is closer to the second unit it is better to use Y = tan(Φ) * X. If, for example, the wrong solution is used, say, Y = tan(Φ) * X when X is very small, tan(Φ) will be very large and small variations in X will make a huge difference in the result. The alternative is Y = tan(Θ) * (X - F), where tan(Θ) will be F times smaller than tan(Φ) and (X - F) = ~F. This results in a much more stable solution.

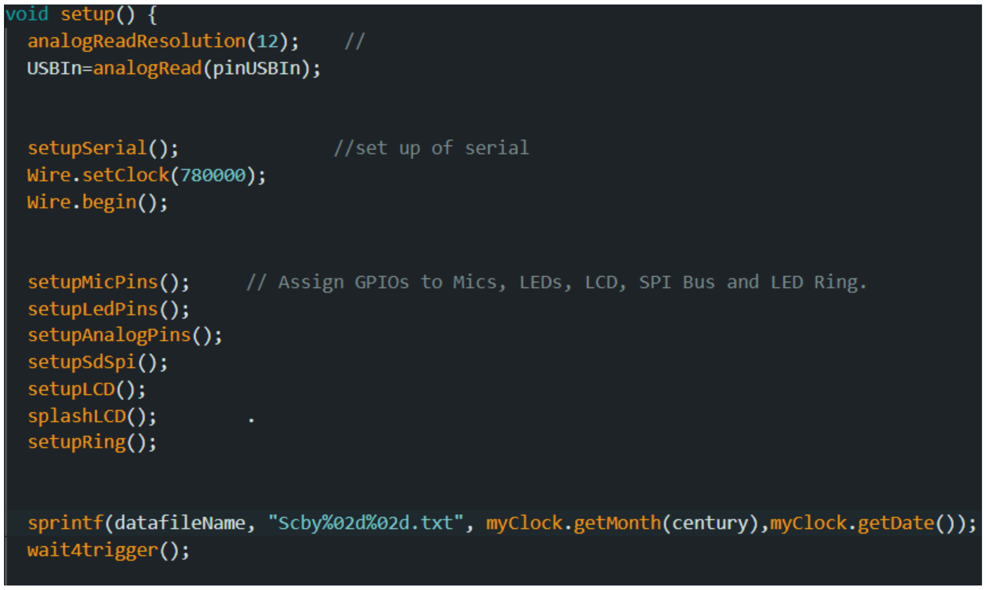
Microphones

SCOOBY contains two types of microphones, four microphones for direction (0-3 on the top of the unit) and one microphone for waveform capture (in the center of the circle, detail below). The four directional microphones are used to determine the direction of a gunshot relative to Mic0, as demonstrated in the diagram above. The fifth microphone in the center collects samples of the waveform in a smoothly changing analog signal which is then converted to digital. This arrangement guarantees that a trigger mic will be triggered before the pressure wave reaches the center mic. That way the entire waveform is captured. Conversion factors (Pascals to digital code) are shown below.



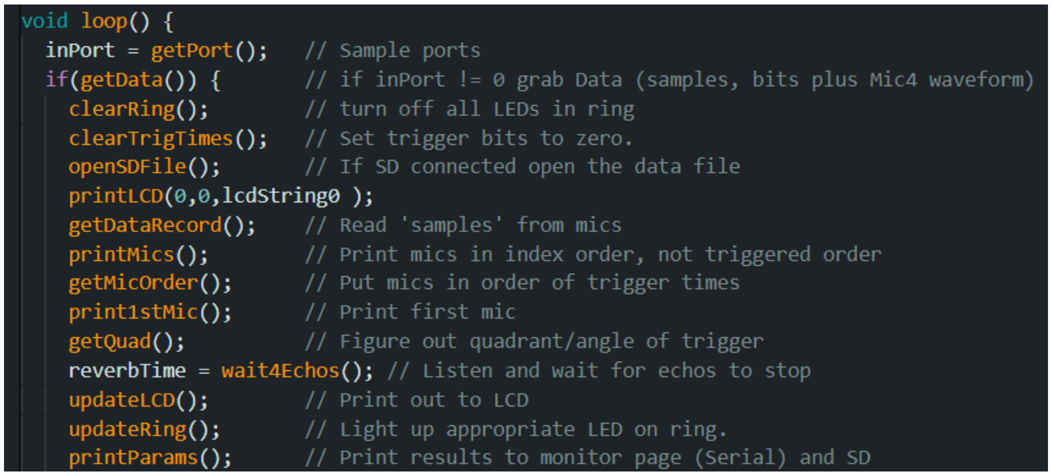
Main Code

SCOOBY's Code is written in C++ language. There are two main parts of SCOOBY's code: the first part being the Main Code where all the Setup functions and Loop functions are contained. The second part of the code being the library, SCOOBY's library is where all the necessary functions and variables are defined. The Setup function in the code is where serial communication is set up and assigns the pins that are being used. It also creates a file for SCOOBY to record its data in.



```
void setup() {
  analogReadResolution(12);    //
  USBIn=analogRead(pinUSBIn);


  setupSerial();             //set up of serial
  Wire.setClock(780000);
  Wire.begin();


  setupMicPins();     // Assign GPIOs to Mics, LEDs, LCD, SPI Bus and LED Ring.
  setupLedPins();
  setupAnalogPins();
  setupSdSpi();
  setupLCD();
  splashLCD();       .
  setupRing();


  sprintf(datafileName, "Scby%02d%02d.txt", myClock.getMonth(century),myClock.getDate());
  wait4trigger();
```

While the Loop function activates after the setup is complete, the loop runs till it detects a loud sound/trigger. When SCOOBY hears a trigger, it will immediately record that data and calculate the angle of the sound. Then proceed to run the loop function all over again, looking for another loud sound to detect.

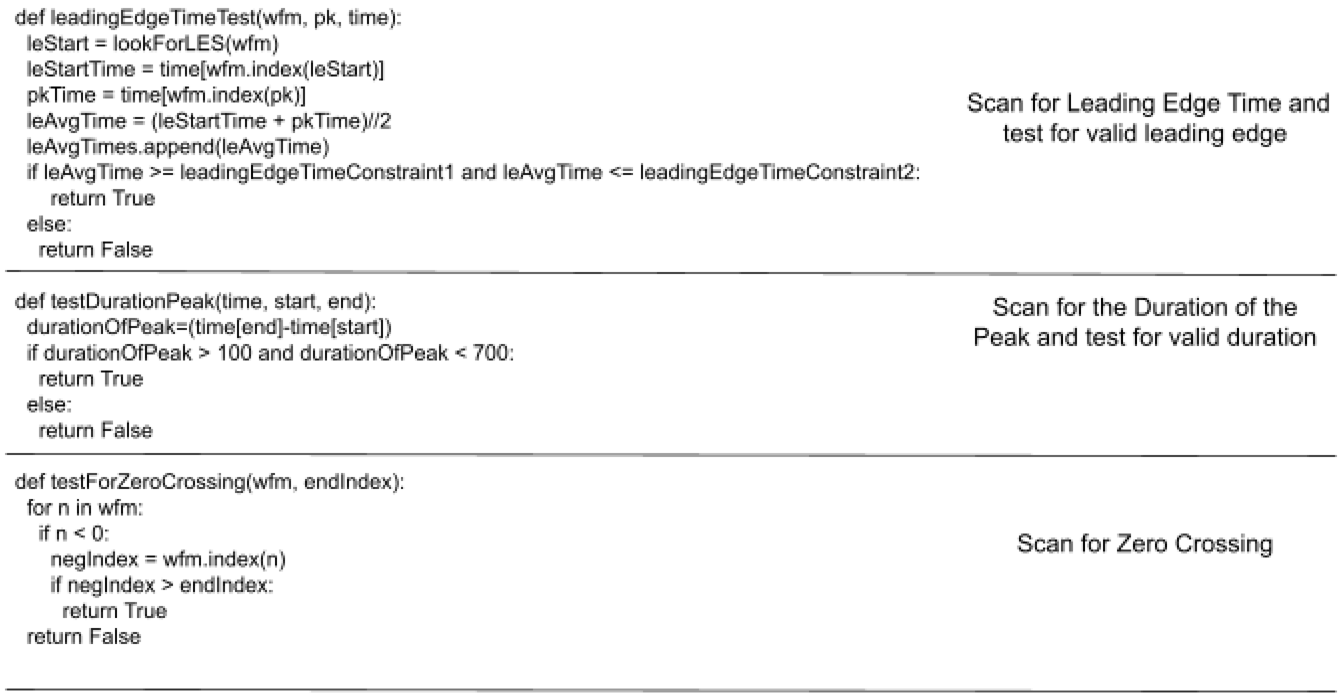


```
void loop() {
  inPort = getPort();   // Sample ports
  if(getData()) {       // if inPort != 0 grab Data (samples, bits plus Mic4 waveform)
    clearRing();        // turn off all LEDs in ring
    clearTrigTimes();   // Set trigger bits to zero.
    openSDFile();       // If SD connected open the data file
    printLCD(0,0,lcdString0 );
    getDataRecord();    // Read 'samples' from mics
    printMics();        // Print mics in index order, not triggered order
    getMicOrder();      // Put mics in order of trigger times
    print1stMic();      // Print first mic
    getQuad();          // Figure out quadrant/angle of trigger
    reverbTime = wait4Echos(); // Listen and wait for echos to stop
    updateLCD();        // Print out to LCD
    updateRing();       // Light up appropriate LED on ring.
    printParams();      // Print results to monitor page (Serial) and SD
```

Waveform Matching Code

SCOOBY has code that discriminates between everyday sounds and balloon pops. The code looks at key features of the soundwave and does some calculations to determine if a sound is a balloon pop. The code was first developed in Python to test and develop a working algorithm for analyzing key points in a soundwave. Later it was translated into C++ and further optimized to run faster.

Waveform Matching Python Model Code



```
def leadingEdgeTimeTest(wfm, pk, time):
  leStart = lookForLES(wfm)
  leStartTime = time[wfm.index(leStart)]
  pkTime = time[wfm.index(pk)]
  leAvgTime = (leStartTime + pkTime)//2
  leAvgTimes.append(leAvgTime)
  if leAvgTime >= leadingEdgeTimeConstraint1 and leAvgTime <= leadingEdgeTimeConstraint2:
      return True
  else:
    return False
```

Scan for Leading Edge Time and test for valid leading edge

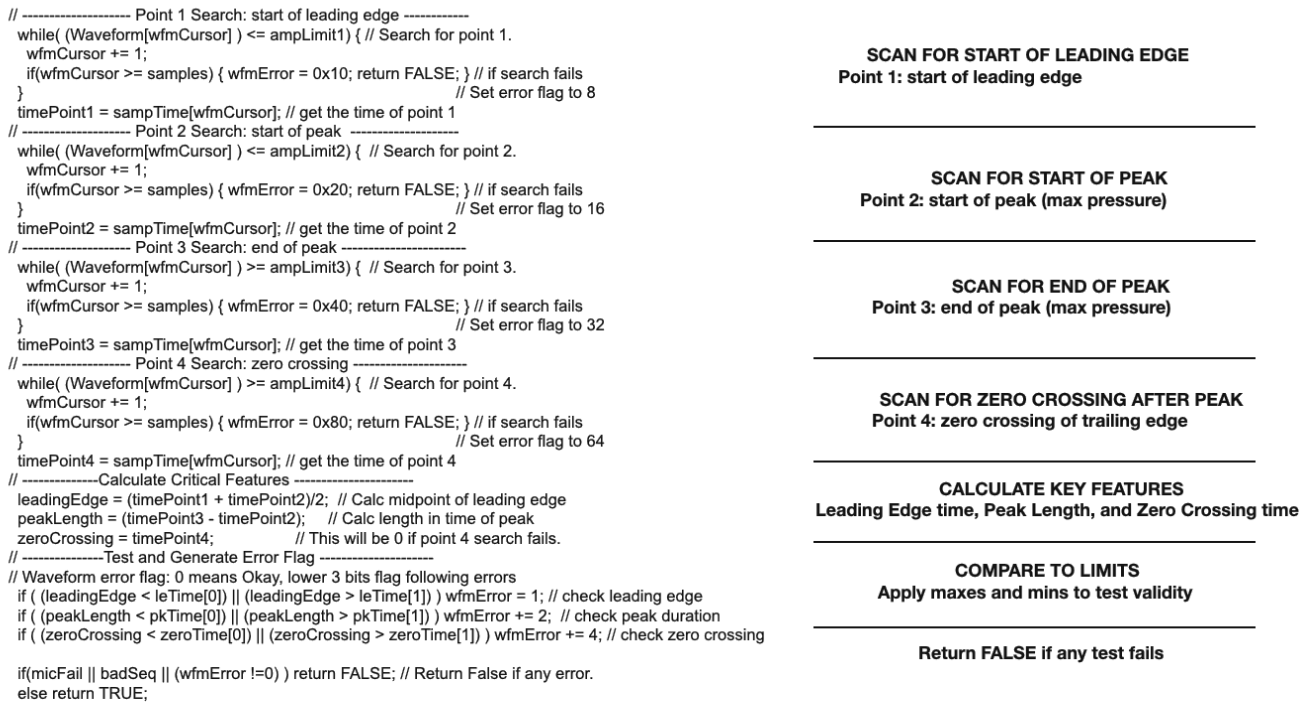
```
def testDurationPeak(time, start, end):
  durationOfPeak=(time[end]-time[start])
  if durationOfPeak > 100 and durationOfPeak < 700:
    return True
  else:
    return False
```

Scan for the Duration of the Peak and test for valid duration

```
def testForZeroCrossing(wfm, endIndex):
  for n in wfm:
    if n < 0:
      negIndex = wfm.index(n)
      if negIndex > endIndex:
        return True
  return False
```

Scan for Zero Crossing

Waveform Matching: Scooby C++ Code



```
// -------------------- Point 1 Search: start of leading edge ------------
  while( (Waveform[wfmCursor] ) <= ampLimit1) { // Search for point 1.
    wfmCursor += 1;
    if(wfmCursor >= samples) { wfmError = 0x10; return FALSE; } // if search fails
  }                                                                // Set error flag to 8
  timePoint1 = sampTime[wfmCursor]; // get the time of point 1
// -------------------- Point 2 Search: start of peak  --------------------
  while( (Waveform[wfmCursor] ) <= ampLimit2) {  // Search for point 2.
    wfmCursor += 1;
    if(wfmCursor >= samples) { wfmError = 0x20; return FALSE; } // if search fails
  }                                                                // Set error flag to 16
  timePoint2 = sampTime[wfmCursor]; // get the time of point 2
// -------------------- Point 3 Search: end of peak ----------------------
  while( (Waveform[wfmCursor] ) >= ampLimit3) {  // Search for point 3.
    wfmCursor += 1;
    if(wfmCursor >= samples) { wfmError = 0x40; return FALSE; } // if search fails
  }                                                                // Set error flag to 32
  timePoint3 = sampTime[wfmCursor]; // get the time of point 3
// -------------------- Point 4 Search: zero crossing ---------------------
  while( (Waveform[wfmCursor] ) >= ampLimit4) {  // Search for point 4.
    wfmCursor += 1;
    if(wfmCursor >= samples) { wfmError = 0x80; return FALSE; } // if search fails
  }                                                                // Set error flag to 64
  timePoint4 = sampTime[wfmCursor]; // get the time of point 4
// --------------Calculate Critical Features ----------------------
  leadingEdge = (timePoint1 + timePoint2)/2;  // Calc midpoint of leading edge
  peakLength = (timePoint3 - timePoint2);     // Calc length in time of peak
  zeroCrossing = timePoint4;                  // This will be 0 if point 4 search fails.
// ---------------Test and Generate Error Flag ---------------------
// Waveform error flag: 0 means Okay, lower 3 bits flag following errors
  if ( (leadingEdge < leTime[0]) || (leadingEdge > leTime[1]) ) wfmError = 1; // check leading edge
  if ( (peakLength < pkTime[0]) || (peakLength > pkTime[1]) ) wfmError += 2;  // check peak duration
  if ( (zeroCrossing < zeroTime[0]) || (zeroCrossing > zeroTime[1]) ) wfmError += 4; // check zero crossing

  if(micFail || badSeq || (wfmError !=0) ) return FALSE; // Return False if any error.
  else return TRUE;
```

SCAN FOR START OF LEADING EDGE
Point 1: start of leading edge

SCAN FOR START OF PEAK
Point 2: start of peak (max pressure)

SCAN FOR END OF PEAK
Point 3: end of peak (max pressure)

SCAN FOR ZERO CROSSING AFTER PEAK
Point 4: zero crossing of trailing edge

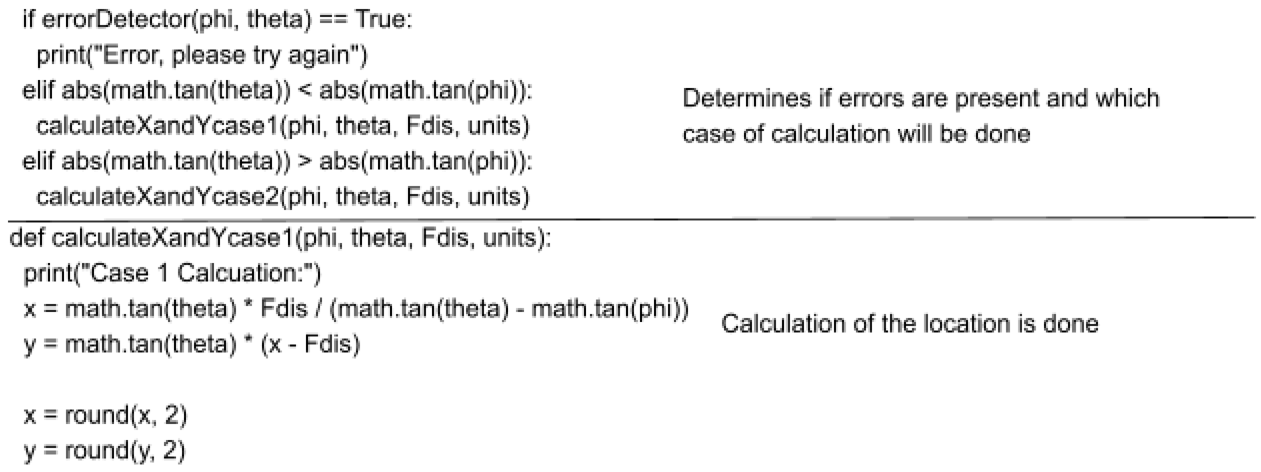
CALCULATE KEY FEATURES
Leading Edge time, Peak Length, and Zero Crossing time

COMPARE TO LIMITS
Apply maxes and mins to test validity

Return FALSE if any test fails

Triangulation Code

SCOOBY's triangulation code is written in Python, the code will be in the main hub. The code reads the serial ports from each unit to get phi and theta. The code code first looks for errors. If there are no errors with the angles, it determines which case is true to calculate the location of the detected sound. Next the calculation of the location of the detected sound is done.



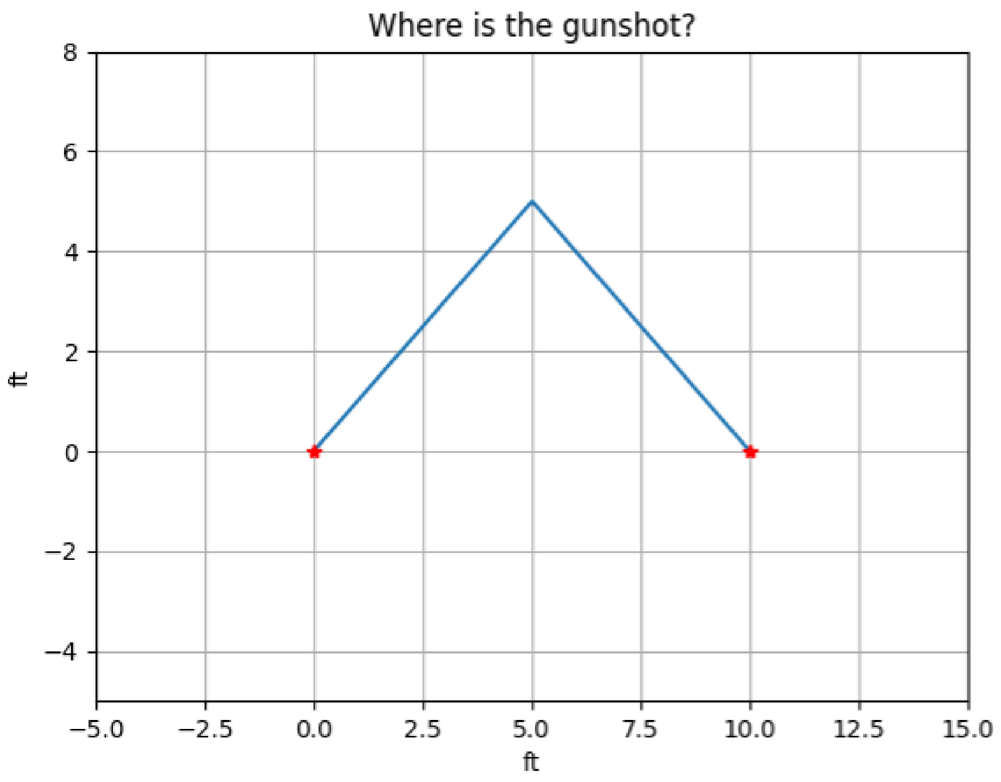
```
if errorDetector(phi, theta) == True:
  print("Error, please try again")
elif abs(math.tan(theta)) < abs(math.tan(phi)):
  calculateXandYcase1(phi, theta, Fdis, units)
elif abs(math.tan(theta)) > abs(math.tan(phi)):
  calculateXandYcase2(phi, theta, Fdis, units)
```

Determines if errors are present and which case of calculation will be done

```
def calculateXandYcase1(phi, theta, Fdis, units):
  print("Case 1 Calcuation:")
  x = math.tan(theta) * Fdis / (math.tan(theta) - math.tan(phi))
  y = math.tan(theta) * (x - Fdis)

  x = round(x, 2)
  y = round(y, 2)
```

Calculation of the location is done

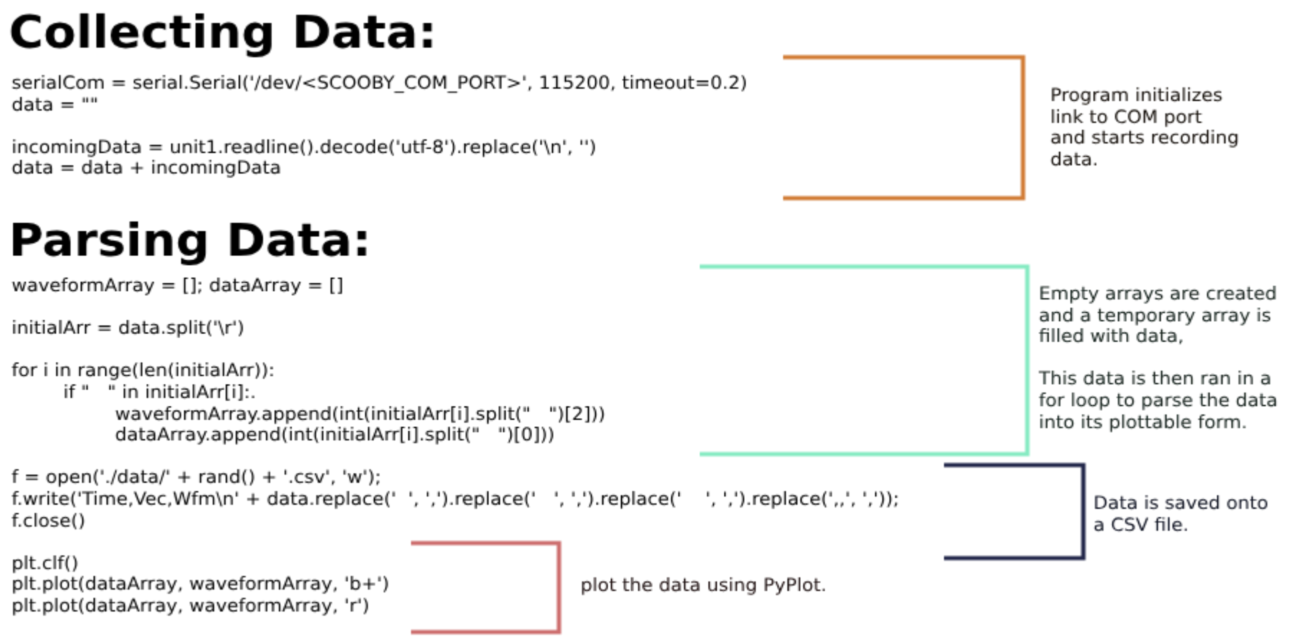
After the code calculates the location of the detected sound. It is then plotted in a graph. The graph represents a room where two units would be placed. There are two red stars on the graph, these are the two units in the room. The two lines intersect at the location of the detected sound. In this case the angle of Phi is 45 and Theta is 145. The graph is produced below.



Machine Learning Algorithm and Data Collection

The system will also have a Machine Learning Algorithm in the future, which will allow predictions to be made with incredible accuracy. To achieve this and triangulation, parsing the data using a main hub is necessary. The hub has its own data collection code, in which it collects, parses and sanitizes data collected from SCOOBY's COM port, it can also plot data using PyPlot, which can generate graphs of the waveform.

Collection Code



Collecting Data:

```
serialCom = serial.Serial('/dev/<SCOOBY_COM_PORT>', 115200, timeout=0.2)
data = ""

incomingData = unit1.readline().decode('utf-8').replace('\n', '')
data = data + incomingData
```

Program initializes link to COM port and starts recording data.

Parsing Data:

```
waveformArray = []; dataArray = []

initialArr = data.split('\r')

for i in range(len(initialArr)):
        if "   " in initialArr[i]:.
                waveformArray.append(int(initialArr[i].split("   ")[2]))
                dataArray.append(int(initialArr[i].split("   ")[0]))
```

Empty arrays are created and a temporary array is filled with data,

This data is then ran in a for loop to parse the data into its plottable form.

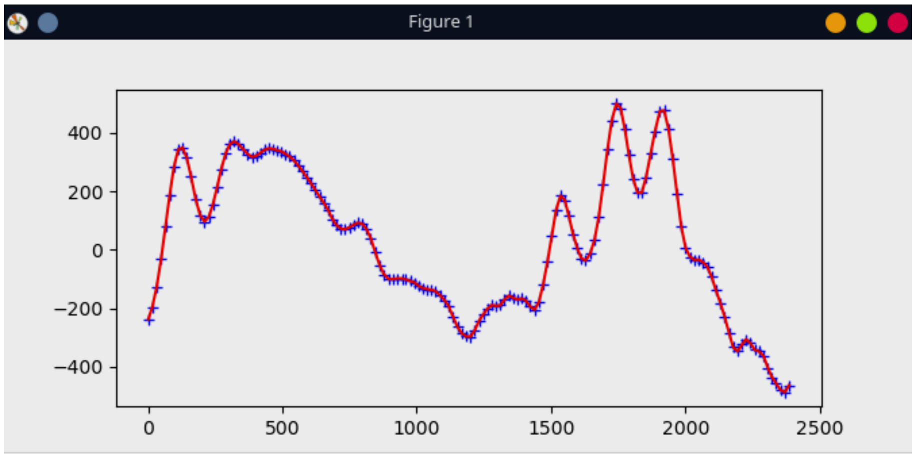
```
f = open('./data/' + rand() + '.csv', 'w');
f.write('Time,Vec,Wfm\n' + data.replace('  ', ',').replace('   ', ',').replace('    ', ',').replace(',,', ','));
f.close()
```

Data is saved onto a CSV file.

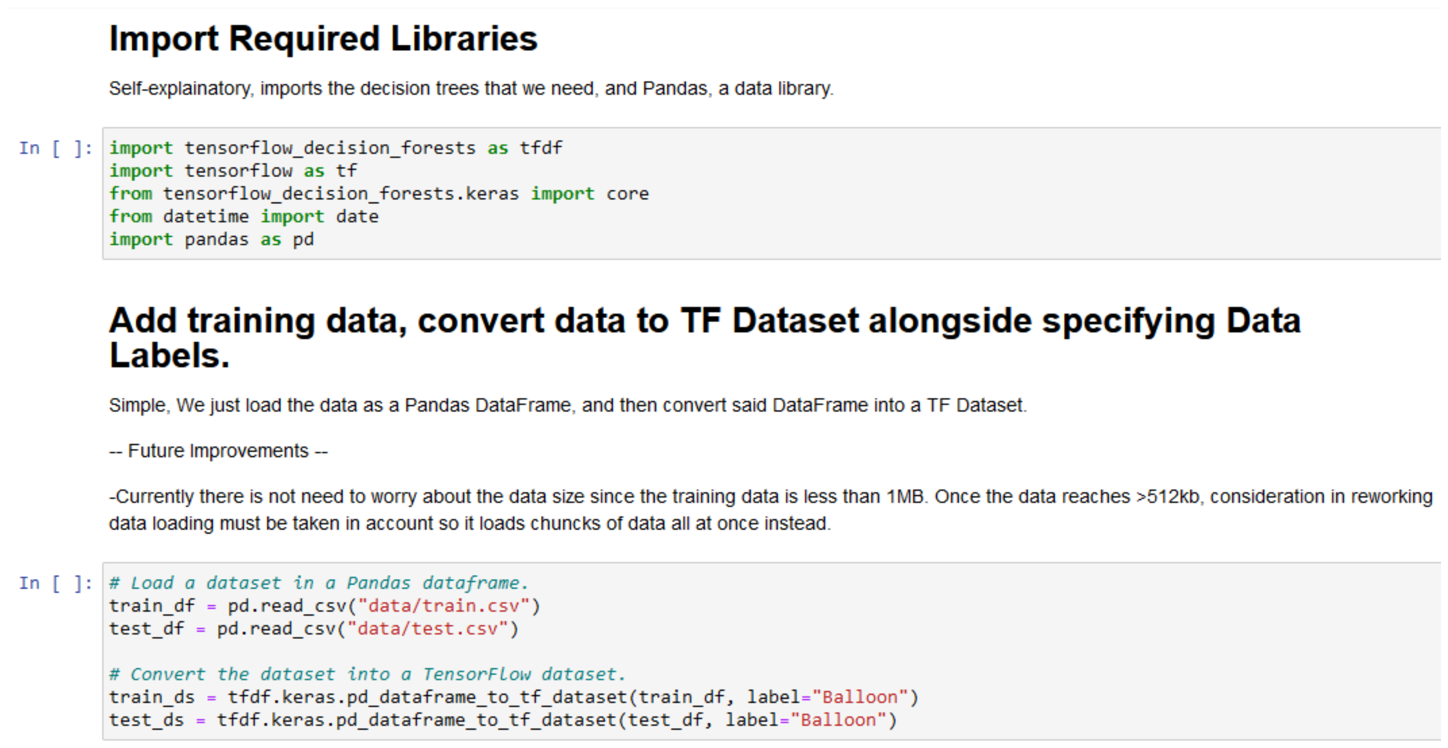
```
plt.clf()
plt.plot(dataArray, waveformArray, 'b+')
plt.plot(dataArray, waveformArray, 'r')
```

plot the data using PyPlot.

Below is a graph of a waveform that the code generated.



Machine Learning Algorithm



Import Required Libraries

Self-explainatory, imports the decision trees that we need, and Pandas, a data library.

```
In [ ]: import tensorflow_decision_forests as tfdf
        import tensorflow as tf
        from tensorflow_decision_forests.keras import core
        from datetime import date
        import pandas as pd
```

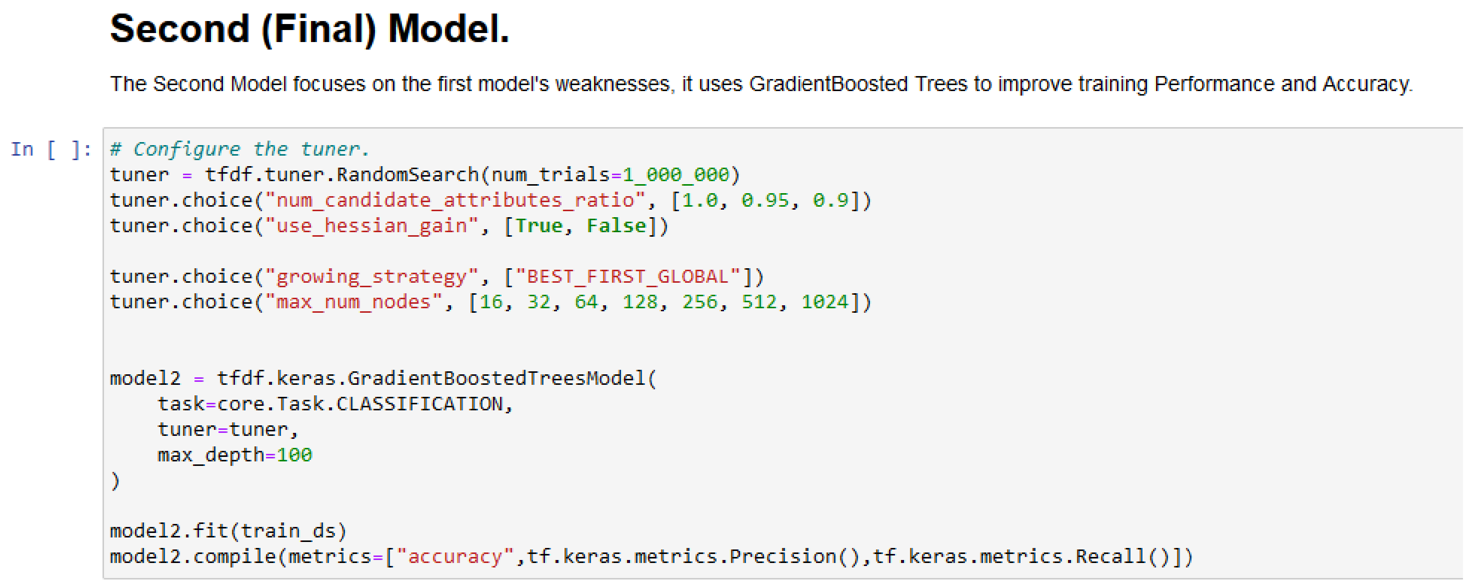
Add training data, convert data to TF Dataset alongside specifying Data Labels.

Simple, We just load the data as a Pandas DataFrame, and then convert said DataFrame into a TF Dataset.

-- Future Improvements --

-Currently there is not need to worry about the data size since the training data is less than 1MB. Once the data reaches >512kb, consideration in reworking data loading must be taken in account so it loads chuncks of data all at once instead.

```
In [ ]: # Load a dataset in a Pandas dataframe.
        train_df = pd.read_csv("data/train.csv")
        test_df = pd.read_csv("data/test.csv")

        # Convert the dataset into a TensorFlow dataset.
        train_ds = tfdf.keras.pd_dataframe_to_tf_dataset(train_df, label="Balloon")
        test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(test_df, label="Balloon")
```

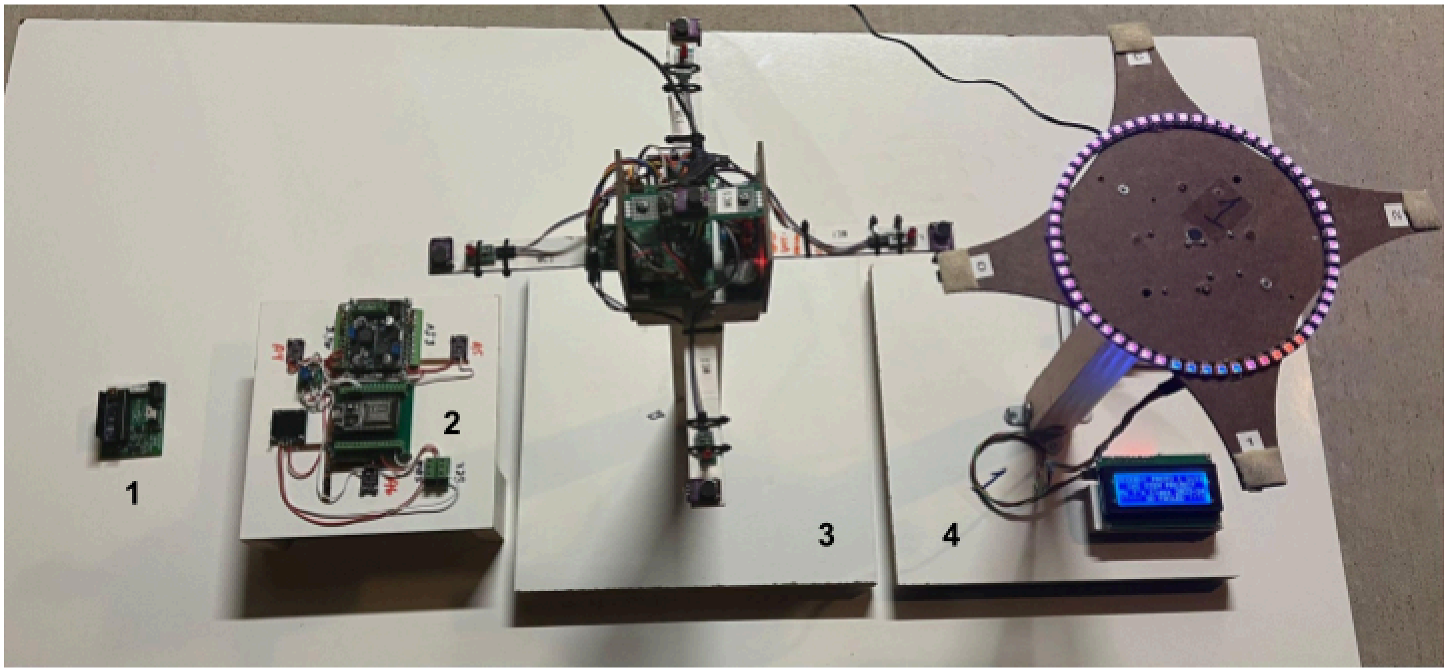


Second (Final) Model.

The Second Model focuses on the first model's weaknesses, it uses GradientBoosted Trees to improve training Performance and Accuracy.

```
In [ ]: # Configure the tuner.
        tuner = tfdf.tuner.RandomSearch(num_trials=1_000_000)
        tuner.choice("num_candidate_attributes_ratio", [1.0, 0.95, 0.9])
        tuner.choice("use_hessian_gain", [True, False])

        tuner.choice("growing_strategy", ["BEST_FIRST_GLOBAL"])
        tuner.choice("max_num_nodes", [16, 32, 64, 128, 256, 512, 1024])


        model2 = tfdf.keras.GradientBoostedTreesModel(
            task=core.Task.CLASSIFICATION,
            tuner=tuner,
            max_depth=100
        )

        model2.fit(train_ds)
        model2.compile(metrics=["accuracy",tf.keras.metrics.Precision(),tf.keras.metrics.Recall()])
```

Prototypes



The project has gone through four different prototypes:

1. SCOOBY originally was on a Arduino Nano board, but it did not have enough memory or speed to complete calculations needed.
2. The second prototype was based on an ESP32. This processor was much faster and had enough memory, but the footprint was too big to fit on the Scooby board.
3. The first full SCOOBY's structure was built with directional microphones out on 'wings' that gave the purest waveform data (as tested against a laboratory grade microphone). This prototype worked, but after many modifications to add features (LCD, Ring Led display, Com port) the unit became unreliable due to constant rework.
4. The fourth prototype was a cleaned up version with minor modifications to the circuit board, a new one piece baffle for the microphones, a box to contain the PCB and wires, and windscreen added over microphones. The new version of the circuit board accommodates the LCD screen (added for more information) and the LED ring (added for easy directional display of the results).

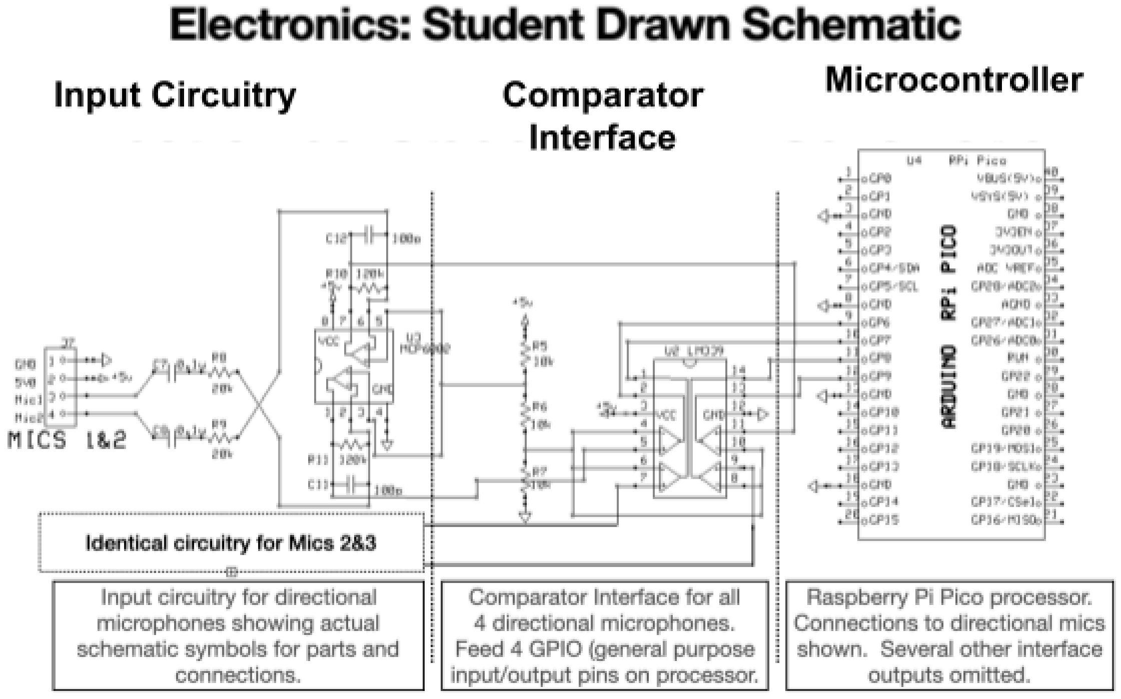
Cost

Electronics	Number Used	Cost each	Total Cost
Mic	15	$1.30	$19.50
SD Drive	3	$6.50	$19.50
Power Input	3	$1.10	$3.30
Regulators	6	$0.65	$3.90
Misc Resistors & Caps	12	$1.24	$14.94
Amplifiers	6	$0.70	$4.20
Comparators	3	$0.50	$1.50
Ring LEDs	3	$12.99	$38.97
LCD Display	3	$9.99	$29.97
Chassis	3	$2.25	$6.75
Real Time Clock	3	$2.38	$7.14
RPi Controller	3	$6.66	$19.98
Printed Circuit Board	3	$35.00	$105
Batteries	3	$1.85	$11.10
Battery Holder	3	$2.25	$6.75
Total Cost: $495.00			

The purpose of this project is to build units as cheaply as possible to keep the cost down for schools so they could implement a system of multiple units of SCOOBY. For this reason, the following inexpensive components were obtained from low cost distributors such as Adafruit, PCBWay, Elegoo, and others.

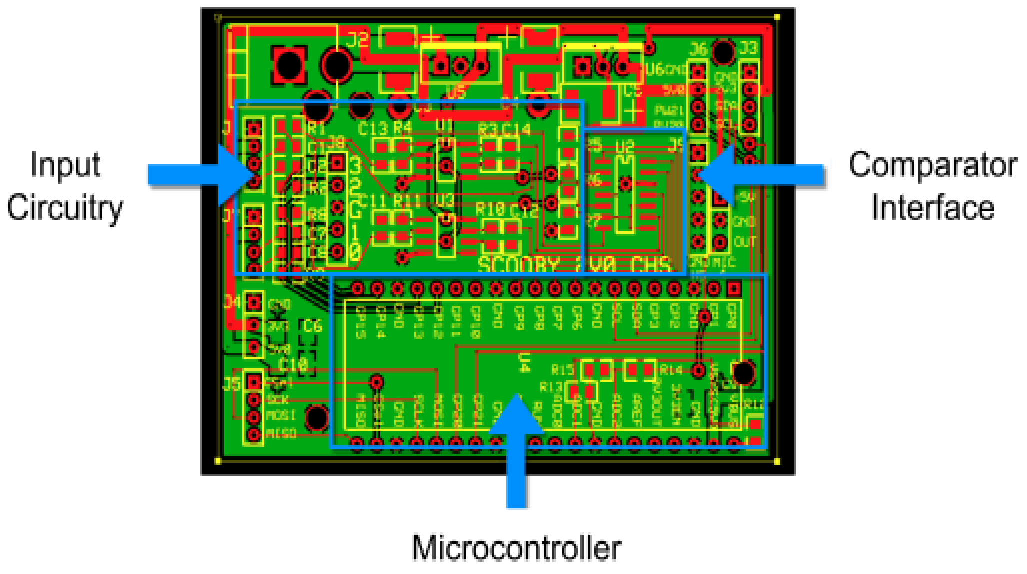
Circuitry

Inside of SCOOBY is a circuit board that the team designed during the summer of 2023. There are three main parts: the input circuitry, the comparator interface, and the microcontroller. The schematic was used as an input to the circuit board layout tool.



Electronics: Student Drawn Schematic
Input Circuitry
Comparator Interface
Microcontroller
MICS 1&2
U3 MCP6002
U2 LM339
U4 RPi Pico
ARDUINO RPi PICO
Identical circuitry for Mics 2&3
Input circuitry for directional microphones showing actual schematic symbols for parts and connections.
Comparator Interface for all 4 directional microphones. Feed 4 GPIO (general purpose input/output pins on processor.
Raspberry Pi Pico processor. Connections to directional mics shown. Several other interface outputs omitted.

Electronics: Student Layout of PCB



Test Model and Evaluation

Safety and Protocol

The testing process included popping large balloons that produced a loud sound that could potentially damage hearing with repeated exposure. To prevent hearing loss the team used ear plugs and stood 20 feet away as an automated system inflated and burst the balloon (BBB and CHAMP). Separate team members were responsible for setting up the layout, managing the automated balloon system and recording data. After each test the SD drive was checked for data integrity.

Testing

The system was tested by popping many balloons in a school's courtyard, hallways and classrooms. At first, the balloons were pumped up manually and then popped with a sharp object such as a screw. Afterwards, the team invested in an automated system which filled the balloons up with air until they blew up. Balloons were used as a substitute for the sounds produced by an actual gun. This is not only for safety reasons, but also because balloons produce a similar decibel range. There was a study done on this subject by University of Alberta Researchers. Heuser Hearing Institute wrote the article “Hearing Loss from Balloons?” using the data from gunshots and balloons inflated until they burst (as done in our tests). The sounds are within 1dB (~10%) of each other.

SCOOBY was placed in the middle of the courtyard. Two team members were responsible for popping the balloons. The tests were conducted at various distances (0-120ft). All the outcomes of each experiment displayed by SCOOBY were read and recorded, and the records of data on the SD card were verified.

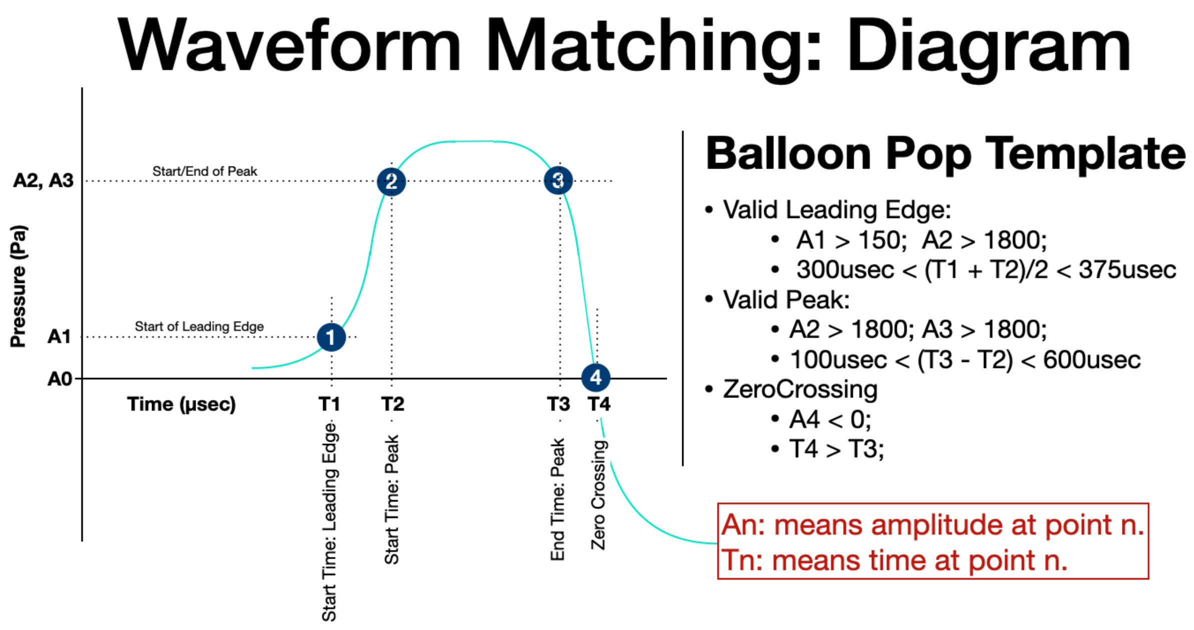
While SCOOBY was being tested, a hypothesis about the balloons was formed. As observed, the balloons dyed in darker colors, such as green and red, took longer to expand. When they finally filled up, they produced a louder sound. Throughout the experiments one thing was mostly consistent. After they “popped”, the way the balloons tore remained relatively similar. It mostly produced two main pieces. One of these pieces consisted of strands of rubber similar to strings. The other was the neck with strands of rubber (the same as in the other piece).

Data Collection

Data collected by SCOOBY was stored into an SD in a TXT file. The file contained three columns of data: Time (µsec), Mic Flags, and Amplitude. The raw data was then imported into a Google Sheet for the purpose of data cleaning and their graphical representation.

Test Results

There are 4 points along the waveform that best characterize the waveform. The points are identified by their amplitude and the time at which that amplitude occurred. For example, point 1 is the first time that the waveform exceeds 150, and point 2 is the first time it exceeds 1800. These limits were tested against the collected data so that actual balloon bursts passed the test, but other sounds like door slams, chair falling, book falling, etc did not.



Waveform Matching: Diagram

Balloon Pop Template

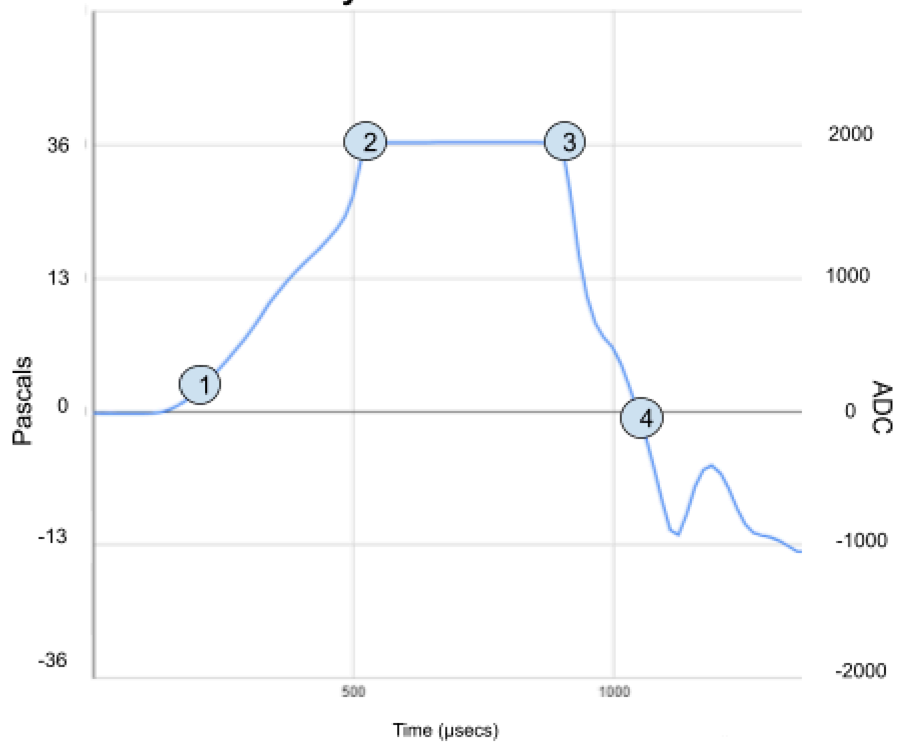
- Valid Leading Edge:
  - A1 > 150; A2 > 1800;
  - 300usec < (T1 + T2)/2 < 375usec
- Valid Peak:
  - A2 > 1800; A3 > 1800;
  - 100usec < (T3 - T2) < 600usec
- ZeroCrossing
  - A4 < 0;
  - T4 > T3;

An: means amplitude at point n.
Tn: means time at point n.

SCOOBY determines four key characteristics of the waveform: leading edge time, peak, peak duration, and the zero crossing time. Between points 1 and 2 is the leading edge of the soundwave, this is calculated with the average times for point 1 and point 2. It must be between 300 µsecs and 375 µsecs. The peak is located at points 2 and 3, the peak must last between 100 µsecs and 600 µsecs. The zero crossing is when the sound has first negative pressure and is at point 4.

After analyzing the data, the conclusion is that as the distance increases the waveform spreads out and gets wider but at extreme distances, the amplitude starts to fall and makes the pulse appear shorter. Down below are examples of data collected:

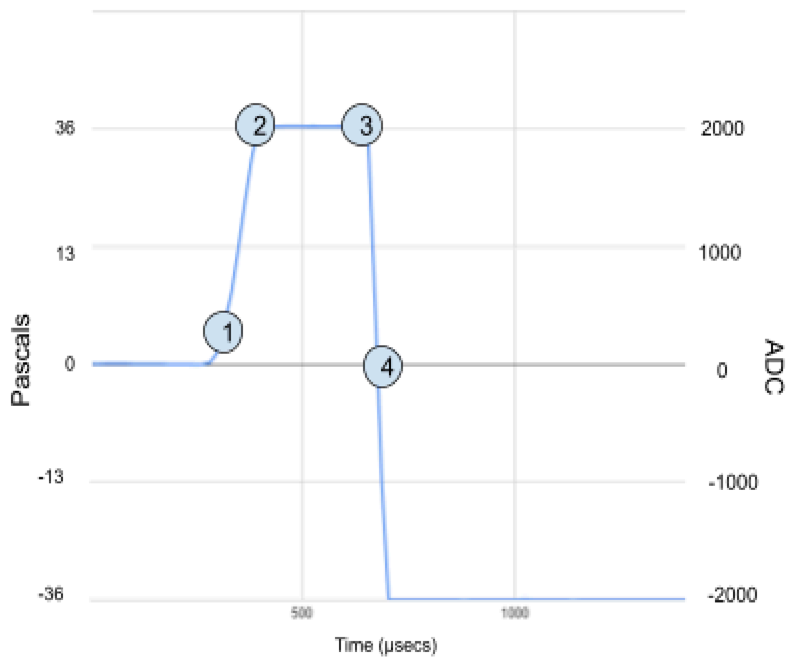
Balloon 120 feet away



Valid Leading Edge:
A1 = 192
T1= 210 usecs
A2 = 1894
T2 = 515 usecs
300 usecs < (210 usecs + 515 usecs)/2 < 600 usecs
300 usecs < 362 usecs < 600 usecs
Valid Peak:
A2 = 1894
T2 = 515 usecs
A3 = 1661
T3 = 915 usecs
100 usecs < 915 usecs - 515 usecs < 700 usecs
100 usecs < 400 usecs < 700 usecs

Zero crossing:
A4 = -189
T4 = 1061 usecs
-189 < 0
T4 > T3

Balloon 20 feet away



Valid Leading Edge:
A1 = 304
T1= 316 usecs
A2 = 2020
T2 = 398 usecs
300 usecs < (398 usecs + 316 usecs)/2 < 600 usecs
300 usecs < 357 usecs < 600 usecs

Valid Peak:
A2 = 2020
T2 = 398 usecs
A3 = 551
T3 = 672 usecs
100 usecs < 672 usecs - 398 usecs < 700 usecs
100 usecs < 274 usecs < 700 usecs

Zero crossing:
A4 = -987
T4 = 688 usecs
-987 < 0
T4 > T3

Triangulation Testing

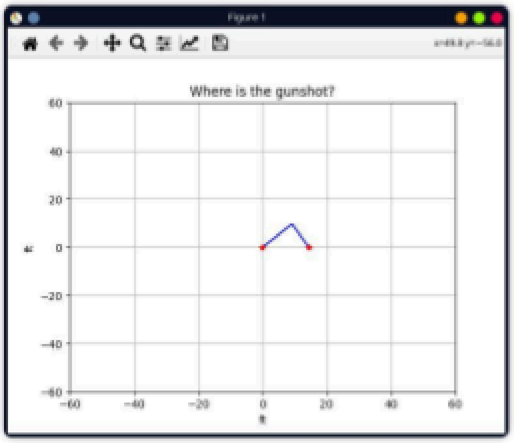
Some functional testing was done on triangulation. The goal of these tests was to see if triangulation is possible with two SCOOBY units connected to a computer running Linux.

There was three tests cases were done for the functional testing:

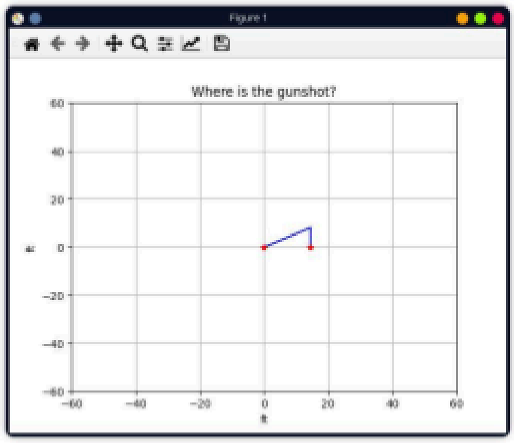
- Case 1: Balloon was popped near between both units, SCOOBY 1's angle was about 45 degrees and SCOOBY 2's angle was about 135 degrees
- Case 2: Balloon was popped near SCOOBY 2, SCOOBY 1's angle was about 45 degrees and SCOOBY 2's angle was about 90
- Case 3: Balloon was popped near SCOOBY 1, SCOOBY 1's angle was about 90 degrees and SCOOBY 2's angle was about 135.

The screenshots produced below are the results of each test case:

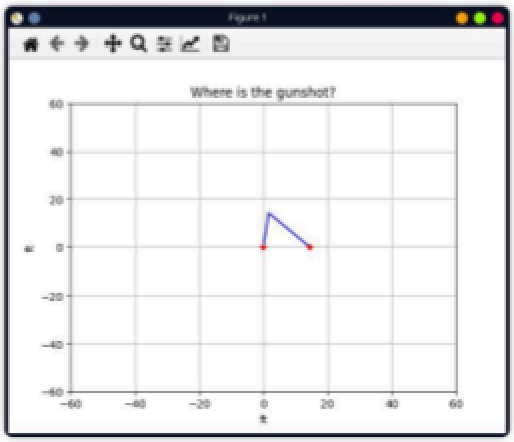
Case 1



Case 2



Case 3



Triangulation with two units is in its preliminary stages, it is important to further test its limit to identify possible errors in calculations.

Strengths

Some strengths that SCOOBY has is that it does accurate calculation of the originating point of a loud sound relative to its microphones. With the use of two units, it will be able to triangulate a loud sound within a room. Also it accurately recognizes a balloon pop, implicating recognition of gunshots in a real event simulation. Each unit of SCOOBY is cost effective and is simple to assemble. Finally it completes all of its completion of all calculations in microseconds (runs 127 thousand times per second). Meaning that in the future it will be able to detect consecutive gunshots.

Weaknesses

Some weaknesses identified while testing is that with the current configuration visually impaired or hearing impaired might not notice device triggering as there is no loud alarm or bright light. There is also a possible vulnerability of false positive detection. More data must be collected to minimize false positives. Also many guns in the market have sound proofing modifications, this may lead to them not being detected by SCOOBY. Only one unit can tell the direction , not the exact location. For triangulation to work there must be at least two units. At this time SCOOBY can’t tell the difference between balloon pops and gunshots (due to safety reasons, SCOOBY has not been tested with actual gunshots). Also SCOOBY currently depends on electricity, meaning if its batteries die the unit is not functional.

Future plans

The future plan with SCOOBY is to test it in its intended configuration. Which is attached to a ceiling panel. Also to help hearing impaired people to know that SCOOBY was triggered, a bright display will be added to inform them. While for the visually impaired a distinct sound will be used to inform them of a trigger. Currently SCOOBY can have a filter so it can only trigger when a balloon is being popped but sometimes it gets triggered with a loud sound similar to a gunshot. To help SCOOBY different balloon pops, gunshots or loud sounds (such as balloon pops, door slams, students dropping items, etc.), it is planned to use a Machine Learning algorithm to differentiate this, preferably using low-cost hardware such as a Raspberry Pi. To be able to save the data that the system collects, a unit will be connected to a central hub via Bluetooth or wired connection. In addition, the hub would have multiple units connected to be able to triangulate sounds to get much more accurate results on where gunshots came from.

Collaboration

The team met regularly on Tuesdays and Fridays to progress on the project. On Tuesdays the team worked on reports/presentations for the competitions. On Fridays the team worked on the project by: collecting and analyzing data, brainstorming new ideas, and gaining experience with the system. During the meetings, the team met with the engineering mentor who guided the development of the project.

Bibliography

"2015 Umpqua Community College Shooting." *Wikipedia*, Wikimedia Foundation, 7 Nov. 2023,en.wikipedia.org/wiki/2015_Umpqua_Community_College_shooting. Date Accessed: November 14 2023

"2018 Santa Fe High School Shooting." *Wikipedia*, Wikimedia Foundation, 4 Nov. 2023, en.wikipedia.org/wiki/2018_Santa_Fe_High_School_shooting. Date Accessed: November 14 2023

Chapman, Isabelle, et al. "Uvalde School Shooter Was in School for up to an Hour before Law Enforcement Broke into Room Where He Was Barricaded and Killed Him." *CNN*, Cable News Network, 26 May 2022, www.cnn.com/2022/05/25/us/uvalde-texas-elementary-school-shooting-wednesday/index.html. Date Accessed: November 14 2023

"Charts & Graphs." *CHDS School Shooting Safety Compendium*, 2 Mar. 2023, www.chds.us/sssc/charts-graphs/. Date Accessed: November 2 2023

"Columbine High School Shootings." *Encyclopædia Britannica*, Encyclopædia Britannica, inc., 18 Sept. 2023, www.britannica.com/event/Columbine-High-School-shootings . Date Accessed: November 14 2023

Craig, Gary. "Rochester Man Shot by Police Sues Cops, City, and Shotspotter." *Democrat and Chronicle*, Democrat and Chronicle, 29 Mar. 2019, www.democratandchronicle.com/story/news/2018/08/30/silvon-simmons-rochester-police-officer-joseph-ferrigno-gun-lawsuit/1119014002/. Date Accessed: November 14 2023

"Estimating Room Impulse Responses from Recorded Balloon Pops", Audio Engineering Society, Conf paper, November 4-7 2010, Jonathan S. Abel et al. Date Accessed: July 21, 2023

"Gun Shot Detection." *Police Surveillance in Chicago*, chicagopolicesurveillance.com/tactics/gun-shot-detectors.html. Date Accessed: October 3 2023

Hauser, Christine, and Anahad O'connor. "Virginia Tech Shooting Leaves 33 Dead." *The New York Times*, The New York Times, 16 Apr. 2007, www.nytimes.com/2007/04/16/us/16cnd-shooting.html. Date Accessed: November 14 2023

Fuel-Admin. "Hearing Loss from Balloons?: Heuser Hearing Institute: Blog." *Heuser Hearing Institute*, 29 Dec. 2017, thehearinginstitute.org/hearing-loss-balloons/. Date Accessed: December 2 2023

Radio, Minnesota Public. "MPR: What Happened at Red Lake?" *News & Features*, 22 Mar. 2005, news.minnesota.publicradio.org/projects/2005/03/redlake/. Date Accessed: November 14 2023

"Sandy Hook School Shooting." *History.Com*, A&E Television Networks, www.history.com/this-day-in-history/gunman-kills-students-and-adults-at-newtown-connecticut-elementary-school. Date Accessed: November 14 2023

"Teen Gunman Kills 17, Injures 17 at Parkland, Florida High School | February 14, 2018." *History.Com*, A&E Television Networks, www.history.com/this-day-in-history/parkland-marjory-stoneman-douglas-school-shooting. Date Accessed: November 15 2023.

"Texas Tower Shooting of 1966." *Encyclopædia Britannica*, Encyclopædia Britannica, inc., 18 Sept. 2023, www.britannica.com/event/Texas-Tower-shooting-of-1966. Date Accessed: November 14 2023

"The Long, Shameful List of Gunfire on School Grounds in America." *Everytown Research & Policy*, 28 Feb. 2023, everytownresearch.org/maps/gunfire-on-school-grounds/. Date Accessed: November 15 2023

“Variations in recorded acoustic gunshot waveforms generated by small firearms”, Acoustical Society of America, January 24 2011, Steven D. Beck et al. Date Accessed: July 20 2023

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

The Oncology Explorer

New Mexico Supercomputing Challenge

Final Report

10 April 2024

School Name: Early College Academy/Justice Code

Team Member(s): Aileen Ukwuoma

Sponsor: Rebecca Campbell

Project Mentor(s): Robert Taylor, Ph.D. and Justin Baca, MD, Ph.D.

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

Table of Contents

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer,

CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

Executive Summary

Chronic kidney disease (CKD) is a condition in which the kidneys progressively lose their ability to eliminate waste from the human body. Such inability can result in comorbidity, or the acquisition of two or more disorders simultaneously. This experiment examined the statistical correlation between cancer, chronic kidney disease, and heavy metals (HM) exposure. We compiled data related to cancer incidence, heavy metal burdens, and kidney dysfunction from the 2017-2018 National Health and Nutrition Examination Survey (NHANES) to create visuals and draw conclusions regarding our experimental questions. We analyzed various NHANES datasets, isolated features of interest related to our research question, determined correlations, and created machine learning models using Python programming and ChatGPT 3.5. Ultimately, we created two computational models predicting cancer incidence and kidney failure. In the future, we can replicate this experiment with a more comprehensive dataset and evaluate the use of our models in medical settings.

Introduction

Problem Statement

Chronic kidney disease (CKD) gradually inhibits the kidney's filtration process, causing an accumulation of fluid and waste in the body. In recent years, scientists have speculated the comorbidity of CKD and ailments such as cancer and heavy metal exposure. The purpose of this experiment is to explore the statistical correlation between cancer, chronic kidney disease, and heavy metals exposure. Another objective is to use machine learning to visualize data and enable predictions about the development of cancer and kidney dysfunction. Concurrently, we evaluate the efficacy of artificial intelligence (version 3.5 of ChatGPT) as a didactic tool in teaching computer science.

We hypothesize that CKD and heavy metals exposure will be correlated with a higher cancer incidence in the 2017-2018 National Health and Nutrition Examination Survey (NHANES) dataset. We also believe that ChatGPT will enable rapid analysis of NHANES datasets in conjunction with Python programming and machine learning.

This project provides novel insights into the link between HM exposure and the development of chronic ailments like cancer and CKD; such a project is especially pertinent to medically underserved regions. The insights obtained from this project will enrich our understanding of comorbidity and offer, through machine learning, a means of predicting one's susceptibility to the diseases in question.

Background Research

The New Mexico Epidemiology Office found that, in 2014, there were 28,473 (14.6%) hospitalizations for chronic kidney disease in New Mexico, a 6.0% increase from the number of CKD hospitalizations in 2013. [1] Furthermore, 15% of individuals across the United States struggle with chronic kidney disease, which may increase the risk of malignancy, or cancer. [2] Studies have found that CKD is a risk factor for mortality in cancer patients. [3] Also, exposure to heavy metals (such as cadmium, lead, and arsenic) can potentially result in CKD and, concurrently, cancer. [4] There is a gap in the scientific literature concerning the statistical correlation between CKD, cancer, and HM exposure and about methodologies to predict these ailments using machine learning. Our project aims to fill this gap.

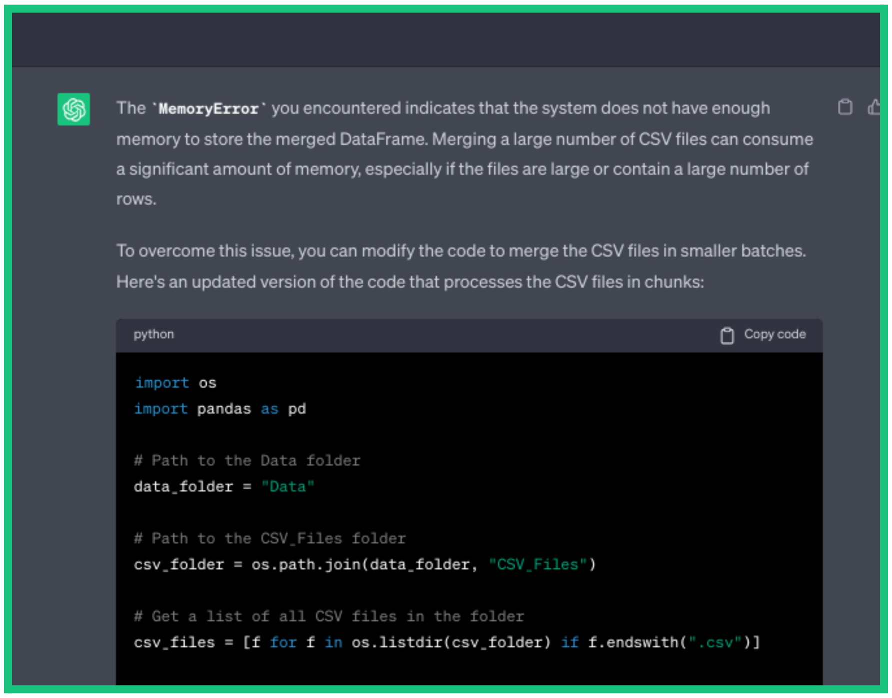
Computational Model

Selection

We selected Python as our primary programming language. Python is useful for machine learning and biostatistical projects such as ours. It allowed us to visualize cancer-related patient data, calculate statistics, determine correlations, and create predictive models using Scikit-Learn. We used Jupyter Notebooks to keep track of our work.

Modifications

While following the general steps of machine learning, we used version 3.5 of ChatGPT as a troubleshooting tool. For instance, while trying to merge crucial NHANES datasets, we encountered "memory" errors due to the large quantity of files. When we consulted ChatGPT 3.5 concerning the issue, the chatbot explained the error and provided alternative means of achieving our goal. This constant feedback allowed us to modify and ameliorate our code.


The `MemoryError` you encountered indicates that the system does not have enough memory to store the merged DataFrame. Merging a large number of CSV files can consume a significant amount of memory, especially if the files are large or contain a large number of rows.

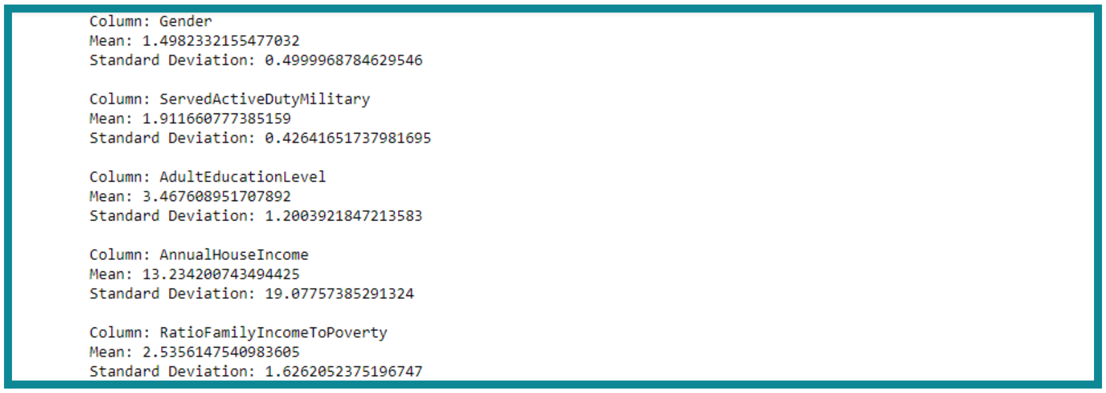
To overcome this issue, you can modify the code to merge the CSV files in smaller batches. Here's an updated version of the code that processes the CSV files in chunks:

```python
import os
import pandas as pd

# Path to the Data folder
data_folder = "Data"

# Path to the CSV_Files folder
csv_folder = os.path.join(data_folder, "CSV_Files")

# Get a list of all CSV files in the folder
csv_files = [f for f in os.listdir(csv_folder) if f.endswith(".csv")]
```

Figure 1. Image Capture of ChatGPT 3.5 Troubleshooting Advice

Visualization

Python allowed us to calculate statistics, determine correlations, and visualize data about cancer incidence, kidney failure, heavy metals exposure, and biochemical markers of the aforementioned conditions. We calculated the means and standard deviations of different columns in our combined NHANES dataset. Python calculated this data for both categorical and quantitative columns, so we had to use discretion in interpreting the platform's statistical results.



```
Column: Gender
Mean: 1.4982332155477032
Standard Deviation: 0.4999968784629546

Column: ServedActiveDutyMilitary
Mean: 1.911660777385159
Standard Deviation: 0.42641651737981695

Column: AdultEducationLevel
Mean: 3.467608951707892
Standard Deviation: 1.2003921847213583

Column: AnnualHouseIncome
Mean: 13.234200743494425
Standard Deviation: 19.07757385291324

Column: RatioFamilyIncomeToPoverty
Mean: 2.5356147540983605
Standard Deviation: 1.6262052375196747
```

Figure 2. Image Capture of Means and Standard Deviations Generated Using Python

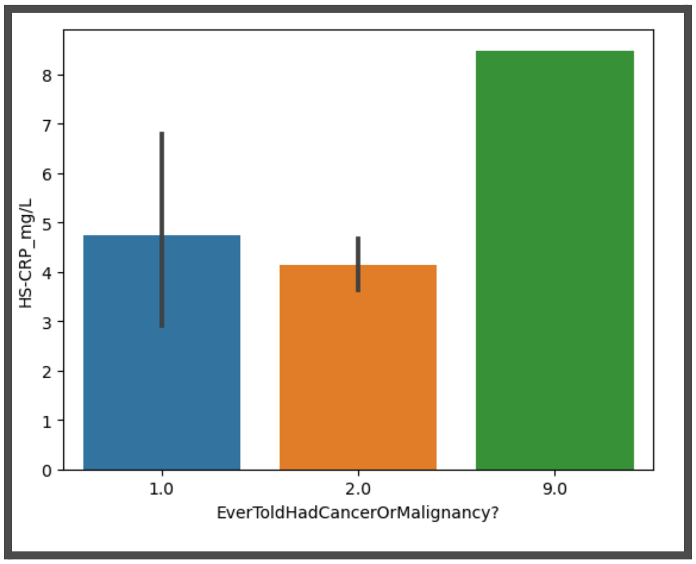


Figure 3. Image Capture of Python-Generated Barplot Comparing High-Sensitivity C-reactive Protein and Cancer Incidence

Limitations

The primary limitation we encountered while completing this project was the high levels of nullity in the NHANES datasets (due to survey respondents' incompletion of certain questions). Often, this forced us to disregard certain columns of data or impute null values with the mean of a column.

Problem Solving Method

Procedure

Given the format of our project, we followed the general methodology of data science and machine learning. The first step in our procedure was data conversion. We analyzed the NHANES 2017-2018 questionnaires and biochemical data, seeking information about cancer, CKD, and heavy metals. The information in these datasets is based on a "SEQN" number, which identifies an individual survey respondent.

We isolated the datasets with features related to our research question, converted them from ".xpt" (SAS, or Statistical Analysis Software, files) to ".csv" (text) files for easier access, and merged these datasets based on the subject identification (SEQN) number. To accomplish this, we consulted ChatGPT for written code that would enable us to combine large datasets.

The second step was exploratory data analysis. Since the NHANES datasets include questionnaires, many survey respondents failed to answer different questions. This resulted in nullity. We examined the structure of the combined datasets, removed duplicate columns, and ameliorated nullity (for instance, by writing code that filled null values or empty slots with the mean of a particular column). Afterward, we visualized our data, creating charts and graphs depicting information for patients with and without cancer.

Variable	Diagnosed With Cancer (n=85)	Not Diagnosed With Cancer (n=763)
Gender (2 levels)	41 (male), 44 (female)	385 (male), 378 (female)

Ever Told Had High Blood Pressure (4 levels)	50 (1), 34 (2), 1 (9)	287 (1), 475 (2), 1 (9)
Ever Told Had High Cholesterol (4 levels)	30 (1), 50 (2), 5 (9)	259 (1), 500 (2), 4 (9)
Dialysis in Past 12 Months? (4 levels)	9 (2)	4 (1), 15 (2)
Ever Told Had Weak or Failing Kidneys? (4 levels)	9 (1), 75 (2), 1(9)	19 (1), 742 (2), 2(9)
Ever Told Had Congestive Heart Failure? (4 levels)	5 (1), 80 (2)	23 (1), 738 (2), 2 (9)
Ever Told Had Coronary Heart Disease? (4 levels)	4 (1), 81 (2)	29 (1), 731 (2), 3 (9)
Ever Told Had Heart Attack? (4 levels)	3 (1), 81 (2), 1 (9)	40 (1), 720 (2), 3 (9)
Do You Smoke Cigarettes? (5 levels)	12 (1), 4 (2), 30 (3)	120 (1), 22 (2), 181 (3)

Table 1. Baseline comparisons of demographic characteristics between respondents with cancer and respondents without cancer.

Thirdly, we explored correlations in the data related to our target variable (cancer incidence). In this step, we encountered difficulty identifying variables specifically pertaining to CKD and HM exposure. The NHANES datasets do not explicitly stipulate whether respondents were diagnosed with CKD. However, these datasets have information about "weak or failing kidneys." For the purposes of this project, we considered this analogous to CKD. Regarding HM exposure, we consulted ChatGPT for assistance in combining columns in our dataset that contained information about the systemic presence of diverse metals into one column entitled "Total Urine Metal Burden (ug/L)." Through Python libraries like Pandas, NumPy, and Matplotlib, we imported the necessary dataframe (as a CSV file) and determined correlations correlations between cancer (recorded in NHANES as "EverToldHadCancerOrMalignancy?"), kidney failure ("EverToldHadWeakOrFailingKidneys?"), urine HM burdens ("TotalUrineMetalBurden_ug/L"), and different biological markers. These correlations were readily displayed in Spearman correlation heatmaps.

The next step in our process was building machine learning models to predict cancer and kidney failure. A crucial component of this step was feature selection. Given the large quantity of information in the NHANES datasets, we needed to narrow down the most important features to improve the predictive capacity of our machine learning (ML) models. For instance, while preparing to create a model predicting cancer, we needed to determine the features most related to cancer incidence. As such, our target variable, or "y," was cancer incidence. Through Python, we used three feature selection methods (Recursive Feature Elimination, SelectKBest, and Feature Importance from a Random Forest Classifier) to determine the features most important to the output variable (cancer incidence). Each feature selection method generated features (which corresponded with a specific NHANES questionnaire question or laboratory test category) that it considered most important when determining cancer incidence.

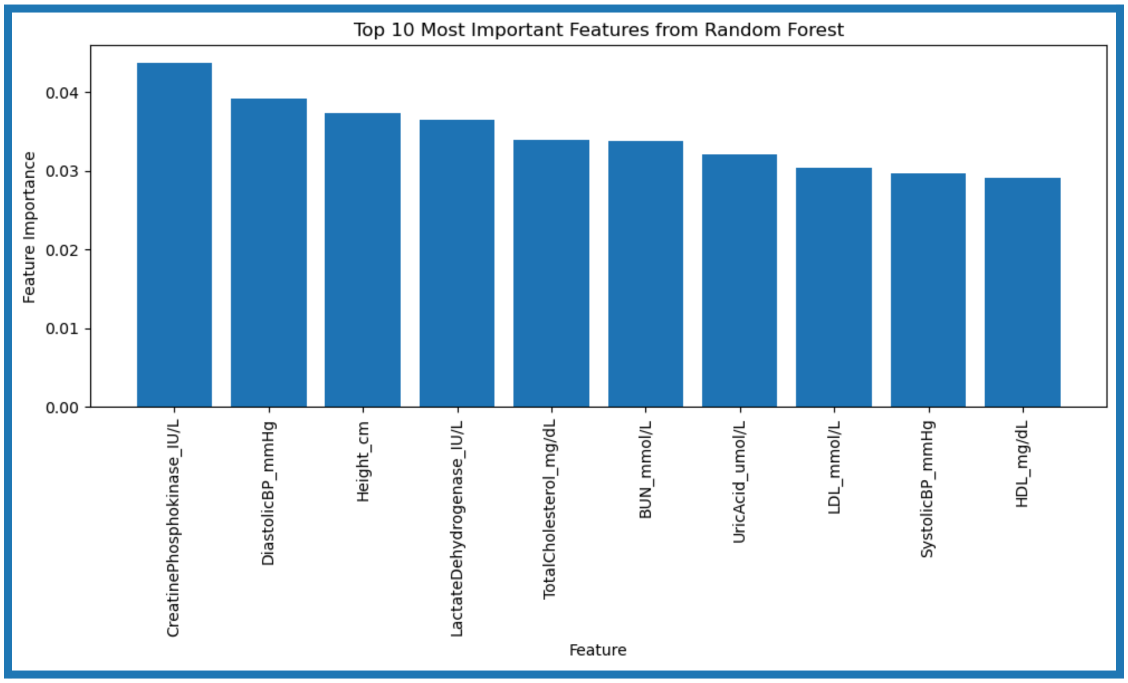
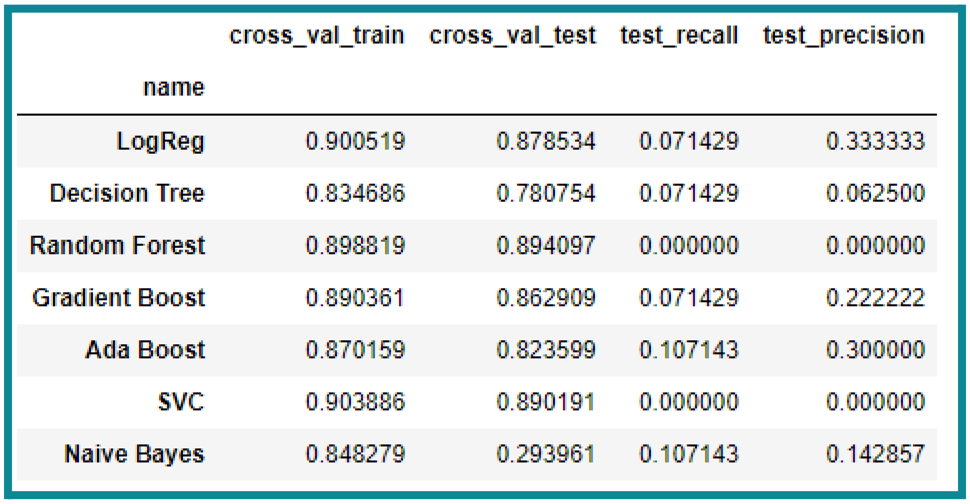


Figure 4. Image Capture of Ten Most Important Features Related to Cancer Incidence, According to the Random Forest Classifier Feature Selection Method

After cleaning the dataset and reducing it to incorporate only features most related to cancer incidence, we used Python to import different classification algorithms (such as logistic regression, decision trees, random forest, and Naive Bayes). Python ranked these algorithms based on their cross-validation, recall, and precision scores. We selected the best-performing algorithm based on these conditions and conducted hyperparameter tuning. In the case of the cancer predictive model, logistic regression had the highest cross-validation, recall, and precision results. We proceeded to perform hyperparameter tuning on this algorithm.



name	cross_val_train	cross_val_test	test_recall	test_precision
LogReg	0.900519	0.878534	0.071429	0.333333
Decision Tree	0.834686	0.780754	0.071429	0.062500
Random Forest	0.898819	0.894097	0.000000	0.000000
Gradient Boost	0.890361	0.862909	0.071429	0.222222
Ada Boost	0.870159	0.823599	0.107143	0.300000
SVC	0.903886	0.890191	0.000000	0.000000
Naive Bayes	0.848279	0.293961	0.107143	0.142857

Figure 5. Results of Classification Algorithm Tests for Cancer Incidence Predictive Model

Essentially, we used statistical metrics to compare the performance of models developed using the aforementioned classification algorithms. For the cancer prediction model, we determined the logistic regression algorithm to be the most reliable. For the kidney failure prediction model, we utilized the Adaptive Boosting algorithm. For both models, we created confusion matrices depicting the accuracy of our models in generating “false positives” or “true positives.”

Conclusion

Results

According to the Spearman correlation heatmaps we developed based on the NHANES dataset (utilizing the Spearman correlation coefficient values), cancer incidence is positively correlated with total protein (0.1), globulin (0.09), alanine aminotransferase (0.09), and urine thallium (0.07). The development of weak or failing kidneys is positively correlated with angina (0.27), high cholesterol (0.13), high-sensitivity C-reactive protein (0.10), and urine cadmium (0.08). The total heavy metal burden is positively correlated with urine creatinine (0.65) and urine albumin (0.44).

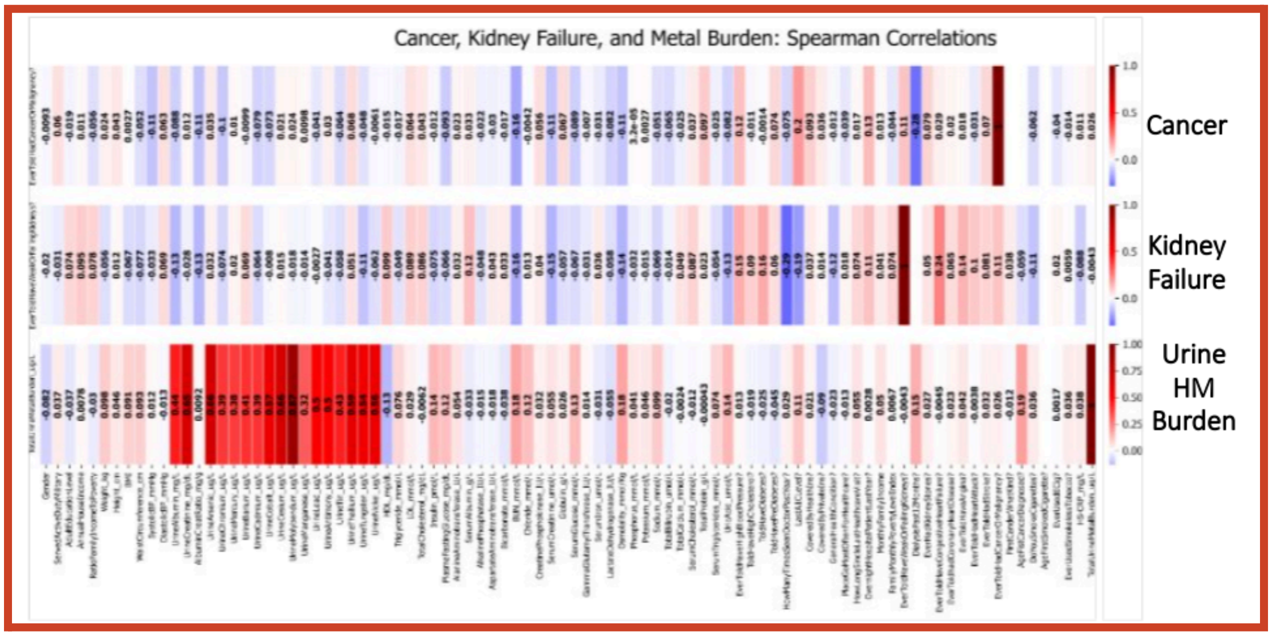


Figure 6. Correlation Heatmaps for Total Metal Burden, Kidney Failure, and Cancer Incidence

As previously mentioned, we developed machine learning models to predict cancer and kidney failure. The logistic regression model we created to predict cancer had better negative predictive power, meaning that it performed better in predicting the absence of cancer than it did in predicting the presence of cancer.

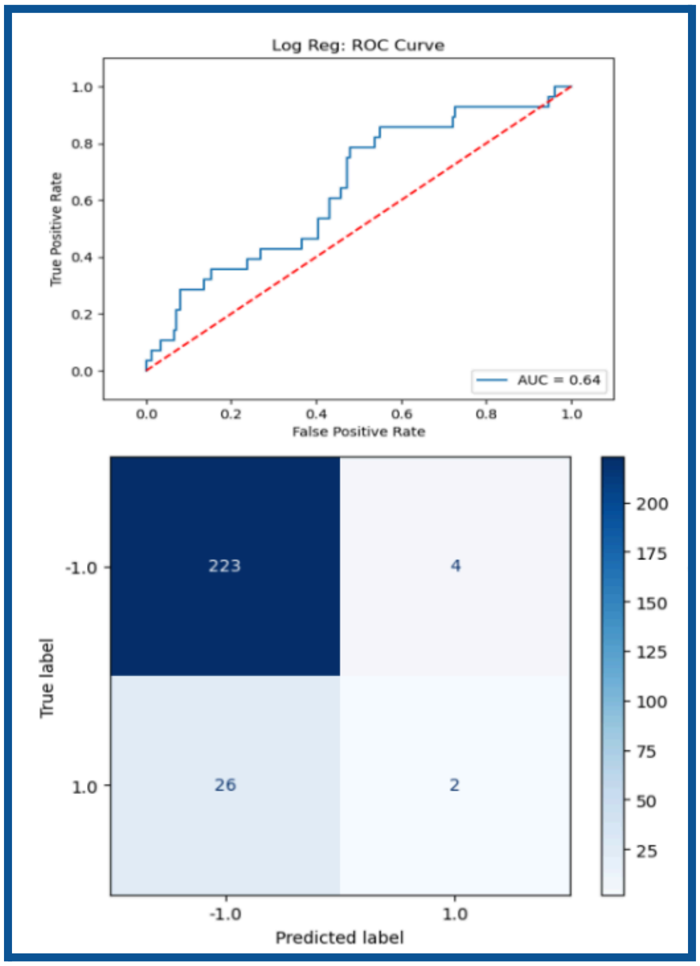


Figure 7. Logistic Regression Receiver Operator Characteristic (ROC) Curve and Confusion Matrix for Cancer Prediction Model

The adaptive boosting model we developed to predict kidney failure appears to function at 100% accuracy. This is highly improbable and an indicator that the model requires optimization.

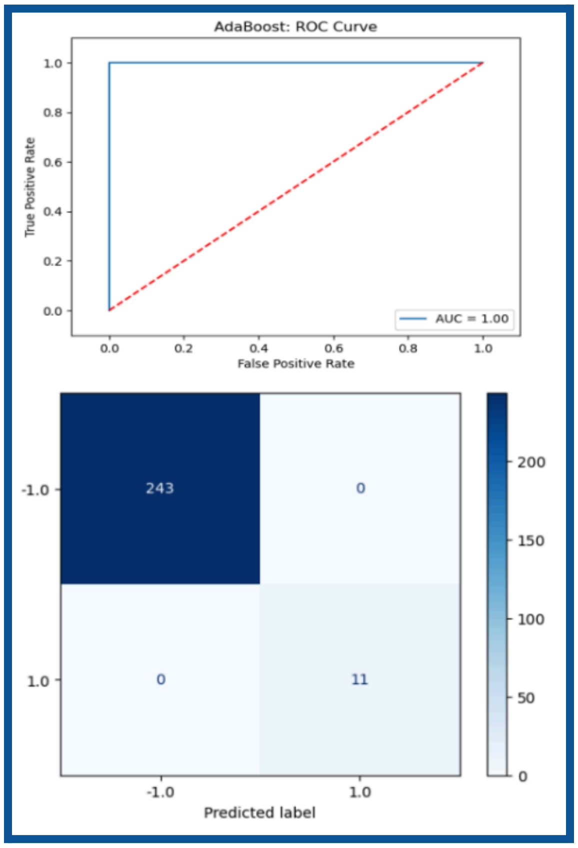


Figure 8. Adaptive Boosting Regression Receiver Operator Characteristic (ROC) Curve and Confusion Matrix for Kidney Failure Prediction Model

Validation and Verification

As discussed previously, our cancer and kidney failure machine learning models underwent validation when we, using Python, split our data into training and test sets. We adjusted the parameters on the training set, assessed performance on the validation set, and evaluated the performance of different classification algorithms to select the most suitable ones. Still, our cancer and kidney failure machine-learning models require further optimization and verification. We will work on this in the near future.

Discussion

Our Spearman correlation heatmap depicted a high correlation between the total metal burden, urine creatinine, and urine albumin levels. The latter two are indicators of kidney failure. As such, this result suggests a correlation between heavy metals exposure and kidney failure. The fact that cancer incidence is positively correlated with alanine aminotransferase (a kidney-and-liver-bound enzyme, the increased presence of which can result in CKD) and urine thallium (a metal) also suggests that cancer is connected to CKD and HM exposure. However, further research into the scientific implications of our statistical results is necessary to ensure the validity of these claims. Our most significant accomplishment in this project was the creation of two machine-learning models predicting cancer and kidney failure.

Future Work

High levels of nullity in the NHANES datasets (due to irresponsive subjects) may have skewed the results of our study. In the future, we intend to refine and optimize our predictive models for cancer and kidney failure while conducting literature searches that corroborate or allow us to verify our research results. We might also replicate our experiment on another, more complete dataset or on other years of the National Health and Nutrition Examination Survey.

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

Acknowledgments

Our involvement in this year's Supercomputing Challenge was made possible with help from many institutions and individuals.

We acknowledge the facilities and resources provided by the National Institute of Health, National Cancer Institute, University of New Mexico Comprehensive Cancer Center (UNMCCC), and the UNM Emergency Medicine Department. These resources were essential for completing this research project. We especially acknowledge the Continuing Umbrella of Research Experiences program, heralded by the UNMCCC.

We appreciate our mentors, Dr. Baca and Dr. Taylor, whose expertise in programming and biological sciences proved invaluable to our project. Dr. Baca and Dr. Taylor receive infrastructure support from the National Center for Advancing Translational Sciences, National Institutes of Health, under grant number UL1TR001449.

Furthermore, we are particularly thankful for the support provided by Ms. Rebecca Campbell from Justice Code and by the Supercomputing Challenge team, including Ms. Patty Meyer, Ms. Celia Einhorn, Ms. Mary Sagartz, and Mr. David Kratzer.

Finally, we extend our gratitude to the judges and scientists associated with the Supercomputing Challenge for their support throughout this endeavor.

Implementing Artificial Intelligence for Bioinformatics and Rapid Analysis of Cancer, CKD, and Heavy Metals in NHANES Datasets by Novice Programmers

References

1. Dirymer V. Chronic Kidney Disease in Persons with Multiple Chronic Diseases, NM, 2014. NMHealth. July 15, 2016. Accessed April 9, 2024. https://www.nmhealth.org/data/view/report/1925.
2. Tendulkar KK, Cope B, Dong J, Plumb TJ, Campbell WS, Ganti AK. Risk of malignancy in patients with chronic kidney disease. PLOS ONE. August 17, 2022. Accessed April 9, 2024. https://doi.org/10.1371/journal.pone.0272910.
3. Guo K, Wang Z, Luo R, Cheng Y, Ge S, Xu G. Association between chronic kidney disease and cancer including the mortality of cancer patients: national health and nutrition examination survey 1999-2014. Am J Transl Res. 2022;14(4):2356-2366. April 15, 2022. Accessed April 9, 2024
4. Jalili C, Kazemi M, Cheng H, et al. Associations between exposure to heavy metals and the risk of chronic kidney disease: A systematic review and meta-analysis. *Critical Reviews in Toxicology*. Published online May 7, 2021:1-30. doi:10.1080/10408444.2021.1891196
5. Nhanes questionnaires, datasets, and related documentation. Centers for Disease Control and Prevention. Accessed April 9, 2024. https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2017.

CARE: Cancer And Radiation Education

New Mexico

Supercomputing Challenge

Final Report

April 10, 2024

Team 42

La Cueva High School

Team Members

Hadwyn Link

Ximena Serna

Iorwen Ouyang

Isaac Chen

Teacher

Jeremy Jensen

Project Mentor

Mario Serna

Table Of Contents

Executive Summary

Introduction: Currently, many aerospace companies are working on interplanetary travel. Since space exploration has a lot of harmful radiation, it is important to prevent exposure whenever possible to avoid the injuries that radiation can cause.

Objective: Our goal is to create a simulation that tracks the relationship between radiation, shielding material, and cancer. From this base, we can run experiments to determine what is the most dangerous radiation and what the most efficient radiation barrier is.

Methods: We are using Geant4 to simulate the radiation bombardment on different materials. The cancer simulation takes the results from Geant to determine how a human would react to the radiation that gets through the barrier. The data from the radiation bombardment and cancer code are presented in a user-friendly format by a front end program.

Results: The radiation simulation demonstrates that most barriers are usually effective at blocking radiation. Gamma particles proved to be the most dangerous, killing the subject in the cancer simulation very quickly. These particles were unable to be stopped by any barrier but were partially blocked by lead. Neon and carbon radiation were especially cancer-inducing, but took a while to deliver a fatal dose to the subject. Fast-moving proton radiation was easily blocked by everything except kapton–a protective film often used in space technology–but was still dangerous if it reached the subject.

Recommendations: Given more time, our project would have included a smartphone application on the app store, an expansion of the materials and radiation types we used to experiment with, more research on how radiation barriers wear out over time, and the effects that radiation would have on satellite parts.

Conclusions: We determined that lead was the most effective material, but was disproportionately expensive compared to the less efficient kapton. Due to its low cost, kapton can afford to be far thicker for the same price. Additionally, we discovered that gamma radiation was incredibly difficult to block by any barrier and also was the most deadly of the tested radiations. However, heavy ions cause the most cancer risk before delivering a lethal dose and are still very dangerous to astronauts for that reason. Our most important part of this project was our bombardment simulation in Geant4 since it took the most effort and provided the base for the rest of the project.

Detailed Report

The Problem:

Currently, many aerospace companies and government agencies are focusing on interplanetary travel. This achievement could lead to major developments in technology and our understanding of the universe, as well as how society functions as a whole. Yet, even with these advantages, space exploration has negative consequences. Along with the negative effects that zero-gravity has on the human body, solar radiation goes unchecked without an atmosphere to block it [1]. Health risks range from decreased nervous system functionality to a significantly increased risk of cancer. However, radiation is not just a problem in space. Even within Earth's atmosphere, there are several instances where more knowledge about preventing radiation exposure is necessary: nuclear reactor meltdowns, radioactive material testing, and UV radiation from our sun are all common sources of radiation that must be approached with safety and exposure prevention in mind.

The Objective:

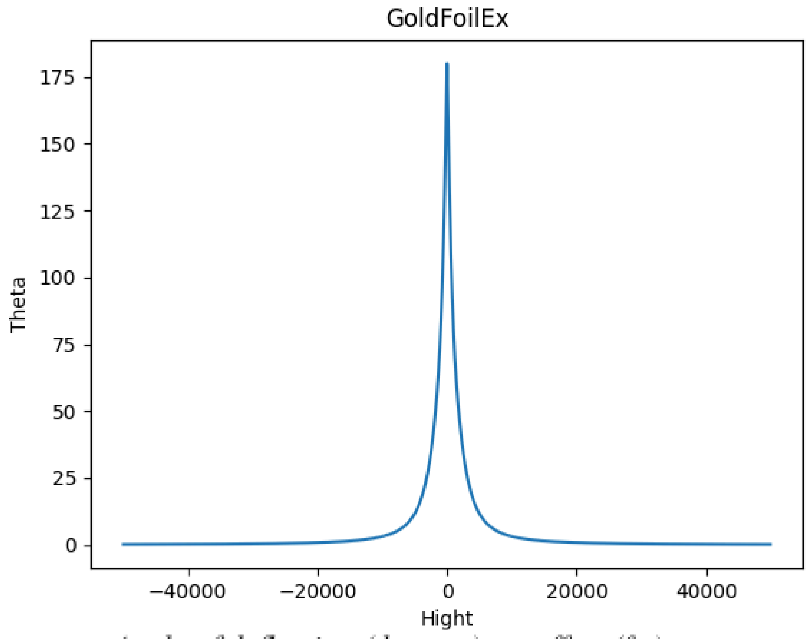
Our objective for this project is to track the relationship between the type of radiation, the materials used to shield it, and the effect that those two factors have on the health of a human behind the shield. Our end product should consist of information about the effectiveness of different materials commonly used for radiation barriers and about the danger that radiation poses to a person–specifically under the conditions of acute radiation exposure. We also want to present these pieces of information in a visually appealing way that can be understood by people who don't have a detailed understanding of the fields of dosimetry or radiation shielding. From this information, we should be able to determine the most effective barrier to keep a human safe for the longest time and determine how dangerous various radiation types are on a biological scale.

The Solution:

Radiation Section

The radiation simulations used Geant4, a CERN-run toolbox for physics simulations [2]. Gears is a specific version of Geant4 with more intuitive syntax. The results were based on cross-sections (the likelihood two materials will interact) for each particle and barrier. If the bombarding object does not have a cross-section, as is the case for beta particles and slow-moving protons, then its energy is not high enough to penetrate the barrier. Because Geant4 is based on cross-sections, the Δt (time step between start and finish) is not calculated for these particles.

To check the accuracy of Geant's cross-sections, a simulation was written to mimic the gold foil experiment and compare the results of the program to the results of the original experiment. The gold foil experiment [3] measured the scattering effect gold foil had when bombarded with alpha particles; in other words, its cross-section [4]. The results from our program matched the results from the original experiment within an acceptable margin of error. Because Geant properly calculated the cross-section in the gold foil experiment, we assume that the accuracy of similar calculations, like the solar wind simulations, is just as accurate.

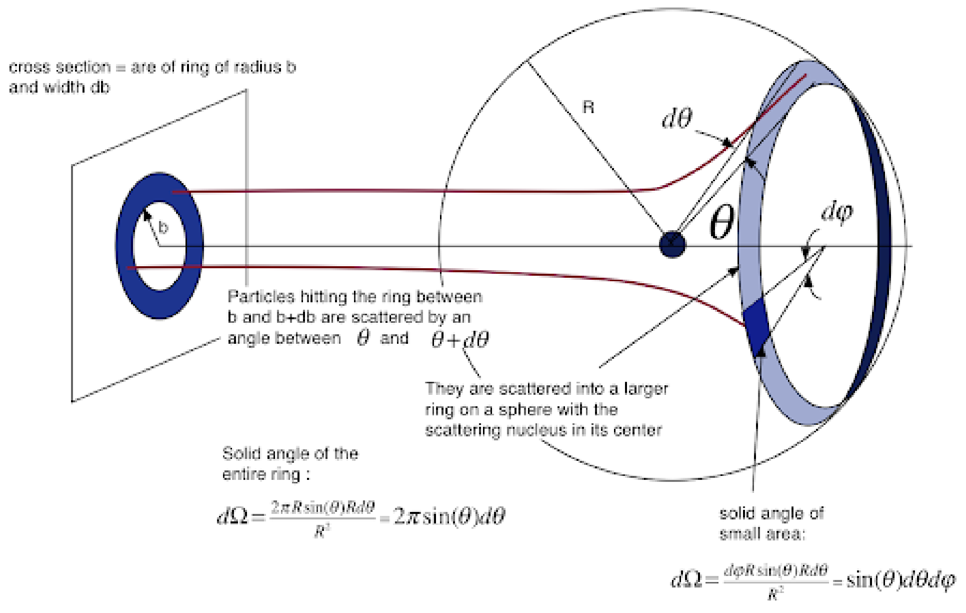


Angle of deflection (degrees) vs. offset (fm)

In order to understand the cross-sections used by Geant, we made a Python program to demonstrate the scattering effect one particle has on the other. This video demonstrates the interaction between an alpha particle and a gold atom similar to that of the gold foil experiment. A similar program was made to demonstrate the relationship between the height change or the angle of approach change and the deflection (scattering) angle. Both of these programs demonstrate a cross-section on a particle scale.

Based on the percentage of the particle's occurrence in the solar wind, the distance, a Python program uses the inverse square law ($I = \frac{W}{4\pi r^2}$) to calculate the Δt with the given number of particles [5]. Due to its speed, each simulation is run with

100,000 particles. The program finds the time it takes for that number of particles to hit the barrier. If the bombarding object does not have a cross-section, as is the case for slow-moving protons, then its energy is not high enough to penetrate the barrier.


cross section = are of ring of radius b and width db
b
R
$d\theta$
θ
$d\varphi$
Particles hitting the ring between b and b+db are scattered by an angle between θ and $\theta+d\theta$
They are scattered into a larger ring on a sphere with the scattering nucleus in its center
Solid angle of the entire ring :
$d\Omega=\frac{2\pi R\sin(\theta)Rd\theta}{R^2}=2\pi\sin(\theta)d\theta$
solid angle of small area:
$d\Omega=\frac{d\varphi R\sin(\theta)Rd\theta}{R^2}=\sin(\theta)d\theta d\varphi$

Cross-section as a measurement of scattering angle and offset(b) [6]

Each of the simulations had a specified .tg (Text Geometry) file that determined the barrier used in the simulation. The barriers used were aluminum, water, lead, concrete, and kapton. The barriers used materials that had the most capability to stop radiation. The thickness of the materials was constant at 50 micrometers, the standard width of a layer of radiation protection. More layers would be used in a real situation.

Each radiation type has two programs: one high energy and one low energy. These energies match the realistic speeds of particles emitted in the solar wind with the high energy matching a solar flare and the low energy matching the lowest possible emission. The exact energy per particle was found using a Python program, which reverse-engineered the electron volts needed to achieve the high and low speeds of the solar wind (300$\frac{km}{s}$ - 500$\frac{km}{s}$) [7]. The program uses the law of kinetic energy (Ke = $\frac{1}{2}MV^2$) to conserve both the mass and speed into the calculation [8]. After converting the resulting joules to electron volts, the energy is in the proper units to be used by Geant4.

The information from each simulation was saved to a CSV(Comma Separated Values) file, which was then parsed by a Python program; its purpose is to properly organize the number of particles that passed, were absorbed, or were reflected by the barrier. The particle type, number of particles, and their energy were then saved to an Excel file.

Cancer Section

In order for a simulation to estimate the health effects of radiation, it must be able to apply the principles of dosimetry to a situation. Dosimetry is the study of radiation protection and the health effects related to it and uses some theories that are crucial for getting this section of the code to work properly. Dosimetry typically breaks down into three distinct sections: Absorbed dose, Equivalent Dose, and Effective Dose. Absorbed dose governs objective health risks, and uses the unit of Grays [9]. There are two main deterministic effects of radiation exposure that the absorbed dose predicts: Acute Radiation Syndrome (ARS) [10] and Cutaneous Radiation Injury (CRI) [11]. These two injuries scale up as radiation exposure increases and are guaranteed to show up on a radiation victim at high doses. Equivalent dose governs the probability of getting cancer on a full-body scale with the unit of sieverts [12], and can also be used as a metric for overall health. This dosage metric scales with the Linear Energy Transfer (LET) of a radiation wave, and the greater the LET the more damage the radiation can do to DNA [13]. Finally, effective dose governs the probability of getting cancer on a tissue-specific scale [9], which is the limit to our level of accuracy. Different tissues have different vulnerabilities to getting cancer, which is important in determining what type of cancer a person will get. There are several different models for how radiation damage scales with dosage, but we are using the linear no-threshold model [14]. This is the easiest model to program since it follows a linear slope of increasing cancer risk, and is one of the leading theories of cancer prediction being used today. If we use these principles of dosimetry, we can ensure that our predictions of cancer and other damages are accurate to the real world.

Quantity	Definition	New Units	Old Units
Exposure	Charge per unit mass of air 1 R = 2.58 x 10^{-4} C/kg	---	Roentgen (R)
Absorbed dose to tissue T from radiation of type R $D_{T,R}$	Energy of radiation R absorbed per unit mass of tissue T 1 rad = 100 ergs/g 1 Gy = 1 joule/kg 1 Gy = 100 rads	gray (Gy)	Radiation absorbed dose (rad)
Equivalent dose to tissue T H_T	Sum of contributions of dose to T from different radiation types, each multiplied by the radiation weighting factor (w_R) $H_T = \Sigma_R\ w_R\ D_{T,R}$	Sievert (Sv)	Roentgen equivalent man (rem)
Effective Dose E	Sum of equivalent doses to organs and tissues exposed, each multiplied by the appropriate tissue weighting factor (w_T) $E = \Sigma_T\ w_T\ H_T$	Sievert (Sv)	rem

Table indicating the units and definitions for the three levels of dosage we are using. [15]

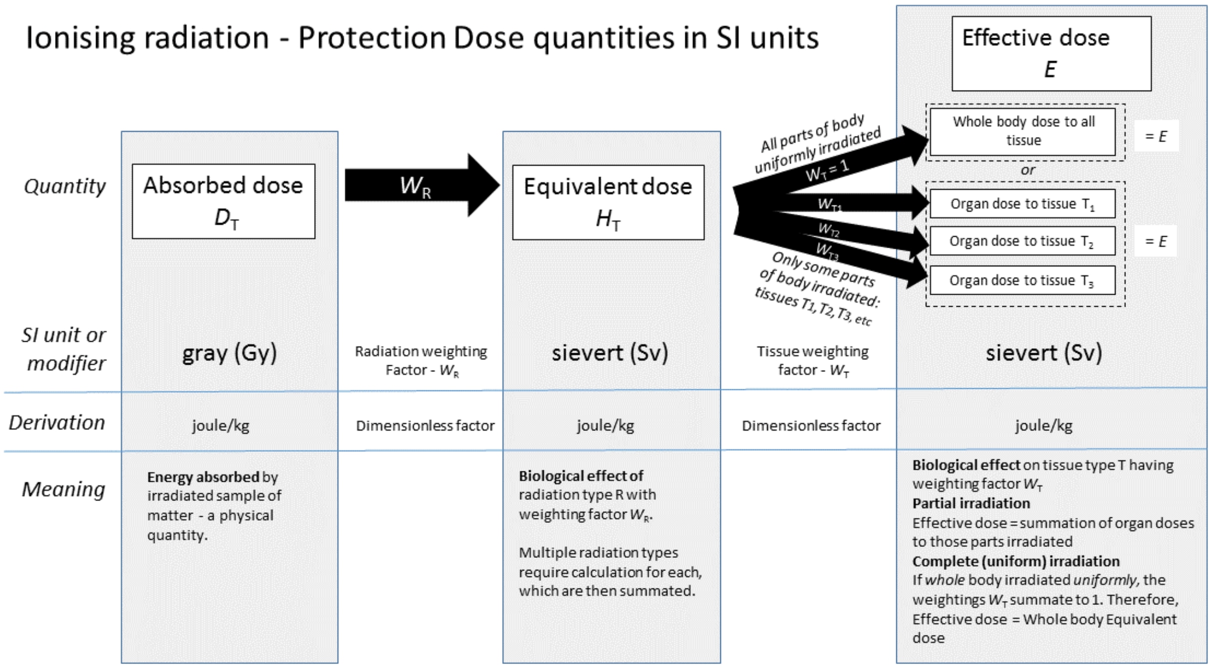
The cancer simulation uses the results from the radiation portion to reach its conclusion. To do this, it takes a target radiation type, barrier type, and radiation speed to look for and then procedurally reads the CSV file until it finds each of those parameters in the same row. From there, it can extract the relevant information for calculating the dosimetry: The two radiation types, the energy per square meter from each radiation type (in KeV/m^2), and the Δt of the radiation bombardment are all saved into variables for easy access later.

To make use of this information, three separate classes define the important factors of the simulation and store crucial information. One is for radiation, which stores its name and the parts of the body that it can hit. One is for deterministic injuries and health effects that can result from radiation exposure. This class stores the injury's name, its current severity, the current symptoms for that level of severity, and a list of all the symptoms of different severities. The final class is for cancers. This class stores the name, what part of the body the cancer affects, the current probability one will develop the cancer given the radiation, and the symptoms that indicate its onset. Three functions are used to translate the units given into the ones used for dosimetry.

To convert electron volts into grays, electron volts must be converted into joules by the scaling factor $(1.6\ *\ 10^{-19})$. Then, the energy must be multiplied by the surface area of the subject to get the net energy that hits the subject, then divided by the subject's mass to get grays: $(G = (J * SA)/M)$ [16]. This is the first function the program runs to acquire the relevant units.

Converting grays into an equivalent dose is based on a scaling factor: multiply the grays calculated from the previous function by the LET of the radiation. This conversion turns the output from grays into sieverts, with one sievert indicating a roughly 5.5% chance of getting cancer [14].

Finally, the third function translates the equivalent dose into the effective dose. This factor, also measured in sieverts, has further weighting to account for tissue vulnerability. Some tissues in the body (e.g. the digestive tract) are more susceptible to ionizing radiation than other tissues (e.g. skin). By accounting for this difference, we can better determine which types of cancer are most likely to appear. For simplicity, we have grouped these tissues into ones relating to the skin, bones, lungs, brain, stomach, and reproductive organs of the body.


Ionising radiation - Protection Dose quantities in SI units
Quantity
SI unit or modifier
Derivation
Meaning
Absorbed dose D_T
W_R
Equivalent dose H_T
Effective dose E
All parts of body uniformly irradiated $W_T = 1$
Whole body dose to all tissue
= E
or
W_{T1}
W_{T2}
W_{T3}
Organ dose to tissue T_1
Organ dose to tissue T_2
Organ dose to tissue T_3
= E
Only some parts of body irradiated: tissues T_1, T_2, T_3, etc
gray (Gy)
Radiation weighting Factor - W_R
sievert (Sv)
Tissue weighting factor - W_T
sievert (Sv)
joule/kg
Dimensionless factor
joule/kg
Dimensionless factor
joule/kg
Energy absorbed by irradiated sample of matter - a physical quantity.
Biological effect of radiation type R with weighting factor W_R.
Multiple radiation types require calculation for each, which are then summated.
Biological effect on tissue type T having weighting factor W_T
Partial irradiation
Effective dose = summation of organ doses to those parts irradiated
Complete (uniform) irradiation
If whole body irradiated uniformly, the weightings W_T summate to 1. Therefore, Effective dose = Whole body Equivalent dose

The relationship between absorbed dose, equivalent dose, and effective dose [17].

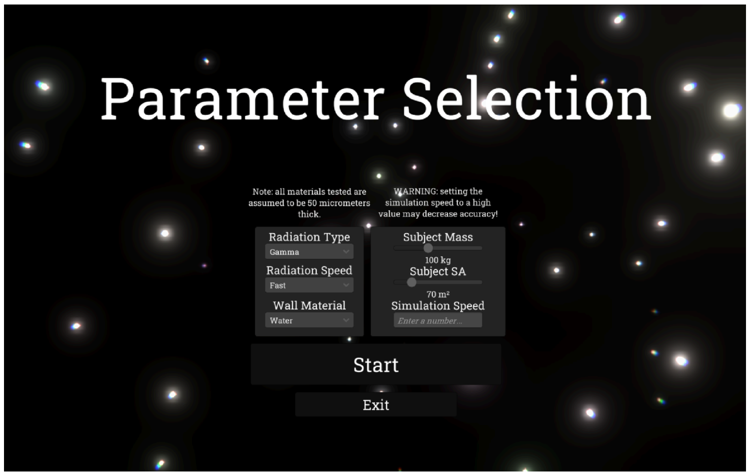
For each iteration of the script, we calculate these three values and add them to a total. The grays are used to determine the severity of the deterministic effects of radiation, such as ARS or CRI. Meanwhile, the effective dose is used to calculate the probability of each type of cancer that we have accounted for, ensuring that each cancer's probability is based on the effective dose of the specific tissue type it affects. In the end, the time elapsed is increased by the Δt of the radiation, and the script is looped.

This loop is repeated until the subject has reached a threshold of radiation exposure where it would be impossible for them to survive: When deterministic effects have reached their maximum severity at 30 grays, or when the total equivalent dose has reached around 50 sieverts. These thresholds pertain only to acute radiation exposure, however; the same dosage over a longer time period may delay symptoms of radiation poisoning as seen in the Plutonium Injection Experiments during the Manhattan Project [18].

Front End Section

In order to easily collect results, as well as fulfill the parameter of having the streamlined, easily comprehensible simulation we defined at the beginning of the project, we created a visualization of the simulation process. While this segment is mainly centered around the output of the cancer portion, it also helps present the findings of the radiation portion in a more visually appealing way than Geant's technical rendering software. As a good starting point for a visualization, we chose Unity Game Engine for its premade 3D graphics engine, easy-to-use particle system to show radiation effects, and intuitive UI(User Interface) design system to display the important values of the simulation.

Unity development is split up into "scenes" – separate zones where you can compile each part of an application, improving performance. For this simulation, we have two scenes: one to set the parameters for the simulation, and one to present the results of said parameters in action.



Parameter Selection

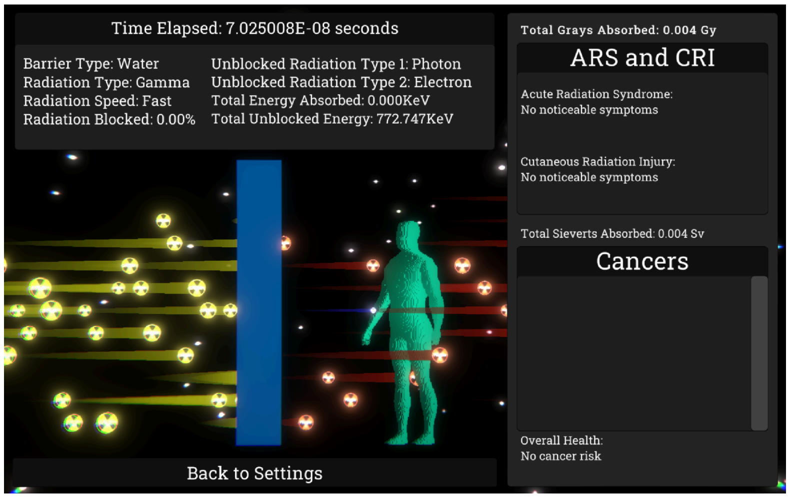
Note: all materials tested are assumed to be 50 micrometers thick.

WARNING: setting the simulation speed to a high value may decrease accuracy!

Radiation Type: Gamma
Radiation Speed: Fast
Wall Material: Water
Subject Mass: 100 kg
Subject SA: 70 m²
Simulation Speed: Enter a number...

Image of the Front End's starting screen.

The starting scene allows the user to modify the parameters of the simulation through a set of dropdown boxes, sliders, and text inputs. With these tools, the user can customize the radiation type, barrier material, radiation speed, and even the physical properties of the subject such as mass or surface area. Note that the selection screen only accounts for results that break through at least one of the radiation barriers—radiation types that are too weak are not included since the results will always be zero. As a bonus, the speed of the simulation can be changed to run simulations more efficiently since most of the timesteps in the radiation section are less than one attosecond (10^{-18} of a second) by default. Once the parameters are set, the application can be run to view results.



Time Elapsed: 7.025008E-08 seconds

Barrier Type: Water
Radiation Type: Gamma
Radiation Speed: Fast
Radiation Blocked: 0.00%
Unblocked Radiation Type 1: Photon
Unblocked Radiation Type 2: Electron
Total Energy Absorbed: 0.000KeV
Total Unblocked Energy: 772.747KeV

Total Grays Absorbed: 0.004 Gy

ARS and CRI

Acute Radiation Syndrome:
No noticeable symptoms

Cutaneous Radiation Injury:
No noticeable symptoms

Total Sieverts Absorbed: 0.004 Sv

Cancers

Overall Health:
No cancer risk

Image of the Front End's simulation screen.

Upon loading the second scene, the settings selected on the starting screen are fed into the cancer section of the project, which uses those settings to generate the outcomes of the simulation. The outputs from the radiation portion are shown on the top panel along with the simulation parameters, and the output from the cancer portion is displayed on the side panel. Through this organization, the UI easily communicates the more important project outputs and makes it easier to log the results once the simulation is complete.

The scene also takes steps to ensure that the simulation has proper visuals to correspond with the data shown, adjusting the color of the radiation particles based on the radiation emitted in the data, along with the emission rate of each radiation type based on how much energy makes it through the barrier. The color of the barrier is also adjusted to indicate the barrier material. Combined with the format of the UI, we have a user-friendly way to present the data we generate.

The Results:

Barrier Effectiveness:

Every type of radiation penetrated the kapton barrier, proving it to be the weakest barrier tested (Refer to Appendix A for raw data). Gamma particles penetrated all the provided barriers with varying success. Lead demonstrated the most resistance to gamma radiation, followed by aluminum, concrete, water, and kapton. For every instance of gamma radiation, the majority of the remaining particles were beta particles (electrons). This results from the photoelectric effect, in which the bombarding gamma particles knock an electron off of its orbital on the barrier. Gamma radiation, therefore, has a higher health consequence from resulting beta particles than it does with resulting gamma particles.

The beta particles and slow protons are slow enough that they do not have a documented cross-section. Consequently, we assume the bombarding particle is absorbed and does not make it past the barrier.

Subject Longevity:

Gamma was the most destructive radiation type by far. Although it had a low LET, gamma rays had significantly more overall energy and particles getting through the barrier. Gamma radiation caused every type of cancer due to its piercing quality and ended the simulation in approximately 5 milliseconds using fast particles, at which point the subject had received a lethal dose. When the particles moved slowly, the subject reached a lethal dose of approximately 2.8 seconds with most barriers. Only lead shielded the subject effectively for 21 seconds. The simulation ended with a 53% chance for digestive cancer at maximum and a low of 1.7% chance for skin cancer. An exception to this rule was using lead shielding, which resulted in the lowest probability of digestive system cancers at a 26.4% chance.

Carbon ion radiation was blocked entirely by every barrier except kapton. With fast-moving radiation the subject received a lethal dose in .65 seconds, having absorbed 2.5 grays and 50 sieverts. The subject received an 88% chance of stomach cancer at maximum and a 2.8% chance of skin cancer at minimum. When the radiation was slow, however, the simulation lasted for 18.5 minutes and ended when the subject had absorbed 14.25 grays and 50 sieverts of radiation.

Similarly to carbon, neon ion radiation couldn’t pierce any radiation barrier except kapton. At high particle speed, the subject reached a lethal dose in .4 seconds, with similar statistics to carbon. At low particle speeds, however, the subject could survive for two full days before reaching a lethal dose with 2.5 grays and 50 sieverts of radiation absorbed.

Proton radiation could not get through any radiation barrier except for kapton, and only at high speeds. The subject lasted for two minutes in this scenario before reaching a dosage of 16 grays or 50 sieverts. This type of radiation only made it through to the skeletal system, with a 35.5% chance of getting skeletal cancer and a 2.8% chance of getting skin-related cancers (Refer to Appendix B for raw data).

The Conclusion:

The radiation portion proved that Gamma radiation was the most dangerous form of bombardment for an interplanetary traveler because of its ability to penetrate all of the barriers. Even though only a fraction of the solar wind is gamma particles, it is the most likely to affect a passenger because of its high wavelength penetrability. Beta particles, even though they didn't penetrate the shield, expose the subject through the photoelectric effect from the gamma rays. Because the gamma rays penetrate every one of our shields at the given thickness, beta particles are guaranteed to generate alongside gamma radiation. After gamma and beta particles, the subject would be at the highest risk of being exposed to heavy ions such as carbon, magnesium, and neon. Any other shield than kapton will protect the passenger from this radiation and bombarding protons.

The most efficient barrier was lead, followed by aluminum, concrete, water, and kapton. Lead, even though it did not successfully block gamma particles, did prove to be the most effective. Kapton was the least efficient at stopping radiation despite its frequent use in NASA spacecraft [19]. However, when considering cost, kapton is the most cost-effective barrier of the materials we experimented with with the exception of water, and lead is the least cost-effective along with concrete [20][21]. There are also other factors to take into account, such as ease of application to space shuttles and the possibility of increasing the layers of the barrier to accommodate the inefficiency of the barriers when blocking certain particles.

Overall, gamma radiation caused the most damage since it had so much total energy compared to the other radiation types. On the other hand, subjects exposed to slow-moving neon radiation survived the longest. However, proton radiation did the least tissue damage, as the subject was only exposed to radiation that could not pass the skeletal system. More efficient gamma protection is clearly needed, since not only is it incredibly lethal if somebody is exposed to it, but it also proves difficult to block; the only material capable of protecting the subject for more than two seconds was lead. More research should be done on heavy ions like carbon and neon as well since they contribute greatly to the chance that an individual will get cancer if exposed.

Over the course of this project, our most significant achievement was the radiation bombardment simulation in Geant. This segment of code required the most overall setup and learning how the library worked before we could use it, and served as the foundation for every other part of the project with the results it generated (See Appendix C for a link to our code repository).

The Recommendations:

There is significant room for expanding the current scope of this project. During the early stages of the development process, we ended up settling on a fixed list of materials and radiation types. By expanding this list, we can improve how comprehensive our final results are.

Additionally, our Front End could be transformed into a smartphone app to both spread information and be more easily accessible to people who want to learn more about radiation shielding and dosimetry.

During our presentations in February, we received some good suggestions from the interviewers concerning how we could expand our scope. This included the idea to elaborate more on how the barriers themselves would hold up over time, as prolonged radiation exposure would eventually break down the efficiency of each barrier. The second recommendation we got was to look into how dosimetry might also apply to how radiation affects technology in space, like satellites or space stations, to see how long these components would last with different protective measures taken.

The Acknowledgements:

We would like to thank our mentor Mario Serna for helping significantly with setting up and teaching us how to use the Geant4 code library. Without him, this project would have taken a lot more time to get up and running. Additionally, we have Regina Hunter to thank for reviewing this paper and making sure that we submitted our best work. Finally, we would like to thank the Supercomputing Staff for a wonderful year of coding. Our team had a couple of rocky points with getting set up and scheduling the interviews for days when our team was available, but the Supercomputing Staff always responded quickly to our requests. While there may not be another year of the Supercomputing Challenge, this year was a great sendoff to a wonderful competition.

Works Cited

[1] Rask, Jon. “Space Faring: The Radiation Challenge.” *NASA*, https://www.nasa.gov/wp-content/uploads/2009/07/284275main_radiation_hs_mod3.pdf?emrc=f9d9bd. Accessed 1 April 2024.

[2] “About Geant4.” *CERN*, https://geant4.web.cern.ch/. Accessed 30 March 2024.

[3]“Rutherford model | Definition, Description, Image, & Facts.” *Britannica*, 13 March 2024, https://www.britannica.com/science/Rutherford-model. Accessed 31 March 2024.

[4] Gomez, Oscar Miyamoto. “Speak physics: What is a cross section?” *Symmetry Magazine*, 24 October 2017, https://www.symmetrymagazine.org/article/speak-physics-what-is-a-cross-section?language_content_entity=und. Accessed 30 March 2024.

[5] “Inverse Square Law - Statement, Formula and Applications.” *BYJU'S*, https://byjus.com/physics/inverse-square-law/. Accessed 30 March 2024.

[6] Iowa State University. “Radiation Safety.” *Nondestructive Evaluation NDE Engineering : Radiation Safety*, https://www.nde-ed.org/NDEEngineering/RadiationSafety/theory/Measures.xhtml. Accessed 2 April 2024.

[7]“PHY5210 W15 Lecture 29.” *High Energy Physics at Wayne State*, http://hep.physics.wayne.edu/~harr/courses/5210/w15/lecture29.htm. Accessed 3 April 2024.

[8]“Real Time Solar Wind | NOAA / NWS Space Weather Prediction Center.” *Space Weather Prediction Center*, https://www.swpc.noaa.gov/products/real-time-solar-wind. Accessed 30 March 2024.

[9]“Kinetic energy.” *Kinetic energy*, https://labs.phys.utk.edu/mbreinig/phys221core/modules/m4/kinetic%20energy.html. Accessed 30 March 2024.

[10] Centers for Disease Control and Prevention. “Acute Radiation Syndrome | CDC.” *Centers for Disease Control and Prevention*, https://www.cdc.gov/nceh/radiation/emergencies/ars.htm. Accessed 2 April 2024.

[11] Centers for Disease Control and Prevention. “Cutaneous Radiation Injury (CRI) | CDC.” *Centers for Disease Control and Prevention*, https://www.cdc.gov/nceh/radiation/emergencies/cri.htm. Accessed 2 April 2024.

[12] United States Nuclear Regulatory Commission. “§ 20.1004 Units Of Radiation Dose. | NRC.gov.” *Nuclear Regulatory Commission*, https://www.nrc.gov/reading-rm/doc-collections/cfr/part020/part020-1004.html. Accessed 2 April 2024.

[13] Haschek, Wanda M., et al., editors. *Haschek and Rousseaux's Handbook of Toxicologic Pathology*. Elsevier Science, 2013. Accessed 2 April 2024.

[14] Connor, Nick. “What is Linear no-threshold model - Definition.” *Radiation Dosimetry*, 14 December 2019, https://www.radiation-dosimetry.org/what-is-linear-no-threshold-model-definition/. Accessed 2 April 2024.

[15] Massachusetts Institute of Technology. “Absorbed Dose.” *DSpace@MIT*, https://dspace.mit.edu/bitstream/handle/1721.1/104092/22-01-fall-2006/contents/lecture-notes/absorbed_dose.pdf. Accessed 2 April 2024.

[16] University of Alabama. *Radiation Safety Initial Training Sessions (RS102) – Module 2: Radiation Safety Calculations*, https://www.uab.edu/ehs/images/docs/rad/CourseMaterialMod2RS102.pdf. Accessed 2 April 2024.

[17] Rolston, Nick. *Human Plutonium Injection Experiments*, 16 March 2015, http://large.stanford.edu/courses/2015/ph241/rolston2/. Accessed 3 April 2024.

[18] Wikipedia. “Effective dose (radiation).” *Wikipedia*, https://en.wikipedia.org/wiki/Effective_dose_(radiation). Accessed 2 April 2024.

[19] “Kapton® Returns to the Moon.” *DuPont*, https://www.dupont.com/blogs/kapton-returns-to-the-moon.html. Accessed 2 April 2024.

[20] “Kapton Film - Dupont KAPTON® Polyimide Film - Order Online.” *Professional Plastics*, https://www.professionalplastics.com/KAPTONFILMPOLYIMIDE. Accessed 2 April 2024.

[21] “Lead Price.” *Daily Metal Price*, https://www.dailymetalprice.com/lead.html. Accessed 2 April 2024.

Centers for Disease Control and Prevention. “Health Effects of Radiation: Health Effects Depend on the Dose.” *Centers for Disease Control and Prevention*, 7 December 2015, https://www.cdc.gov/nceh/radiation/dose.html. Accessed 2 April 2024.

Charles Sturt University. “How much ionising radiation is dangerous? - Research.” *CSU Research*, https://research.csu.edu.au/integrity-ethics-compliance/radiation/forms-templates-profor mas/radiation-life/ionising/how-much. Accessed 2 April 2024.

Cleveland Clinic. “Tumor: What Is It, Types, Symptoms, Treatment & Prevention.” *Cleveland Clinic*, 10 May 2021, https://my.clevelandclinic.org/health/diseases/21881-tumor. Accessed 2 April 2024.

County of Monmouth. *RADIATION HEALTH BASICS.pdf*, https://www.co.monmouth.nj.us/documents/118/RADIATION%20HEALTH%20BASICS.pdf. Accessed 2 April 2024.

Ferrari, Chiara, et al. “Sievert or Gray: Dose Quantities and Protection Levels in Emergency Exposure.” *NCBI*, 8 February 2023, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9959072/. Accessed 2 April 2024.

“GEARS 齿轮组 | Geant4 Example Application with Rich features yet Small footprint.” *physino.xyz*, https://physino.xyz/gears/. Accessed 30 March 2024.

Gilbert, Ethel S. “Ionizing Radiation and Cancer Risks: What Have We Learned From Epidemiology?” *NCBI*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2859619/. Accessed 2 April 2024.

Hansen, Steven. “Concrete Calculator and Cost Estimator.” *Costimates.com*, 9 February 2024, https://www.costimates.com/cost-calculators/concrete-calculator/. Accessed 2 April 2024.

Minow, Joseph I. “Solar Wind as a Space Radiation Environment.” *YouTube: Home*, 9 November 2017, https://ntrs.nasa.gov/api/citations/20210000006/downloads/Minow_Materials%20-%2043rd_COSPAR_Final.pdf. Accessed 30 March 2024.

National Cancer Institute. “A to Z List of Cancer Types.” *National Cancer Institute*, https://www.cancer.gov/types. Accessed 2 April 2024.

“Protecting Yourself from Radiation | US EPA.” *Environmental Protection Agency*, 24 January 2024, https://www.epa.gov/radiation/protecting-yourself-radiation. Accessed 30 March 2024.

“Real Time Solar Wind | NOAA / NWS Space Weather Prediction Center.” *Space Weather Prediction Center*, https://www.swpc.noaa.gov/products/real-time-solar-wind. Accessed 30 March 2024.

Appendix A: Results of Radiation Bombardment

	Aluminum	Water	Concrete	Kapton	Pb
Alpha fast	0	0	0	0	0
Alpha slow	0	0	0	0	0
Gamma fast	100,000	100,000	100,000	100,000	100,000
Gamma slow	97524	99815	97998	100,000	6278
Beta fast	NCS	NCS	NCS	NCS	NCS
Beta slow	NCS	NCS	NCS	NCS	NCS
Carbon fast	0	0	0	84496	0
Carbon slow	0	0	0	155	0
Magnesium fast	0	0	0	82701	0
Magnesium slow	0	0	0	0	0
Neon fast	0	0	0	82844	0
Neon slow	0	0	0	1	0
Proton fast	0	0	0	265	0
Proton slow	NCS	NCS	NCS	NCS	NCS

Number of particles remaining after bombardment

**NCS = No Cross Section*

Appendix B: Results of Cancer Simulation

Rad Type	**Barrier Type**	**Rad Speed**	**Survival Time**	**Cancer Types**	**% chance of cancer (Highest and lowest)**	**Gray and Sievert levels**
Gamma	Water	Fast	0.000495 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Water	Slow	2.734 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Aluminum	Fast	0.000495 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Aluminum	Slow	2.799 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Concrete	Fast	0.000496 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Concrete	Slow	2.783 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Lead	Fast	0.000496 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Lead	Slow	21.894 seconds	All types of cancer	26% digestive, 1.7% skin	30sv, 30gy
Gamma	Kapton	Fast	0.000496 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Gamma	Kapton	Slow	2.726 seconds	All types of cancer	53% digestive, 1.7% skin	30sv, 30gy
Carbon	Kapton	Fast	0.646 seconds	All types of cancer	88% digestive, 2.8% skin	50 sv, 2.5 gy
Carbon	Kapton	Slow	1155.67 seconds	All types of cancer	88% digestive, 2.8% skin	50sv, 14.25 gy
Neon	Kapton	Fast	0.388 seconds	All types of cancer	88% digestive, 2.8% skin	50 sv, 2.5 gy
Neon	Kapton	Slow	169205.4 seconds	All types of cancer	88% digestive, 2.8% skin	50 sv, 2.5 gy
Proton	Kapton	Fast	114.9204 seconds	skin, bone cancer	35.8% bone, 2.8% skin	50sv, 16 gy

Results from the Cancer Effects Portion. Note that only situations in which the subject is exposed to some degree of radiation are on this list.

Appendix C: Link to Code Repository

https://github.com/HadwynLink/SCCTeam42-2023-2024

A Systems Approach to Understanding Artificial Night at Light

Camila Carreon

Santa Fe Prep

Santa Fe, NM

April 2024

Contents

1 Executive Summary

Increasingly, news of climate change and global warming have dominated scholarship and media outlets. Exposure to such topics has warned the public of man-made harm to the environment, like air pollution and greenhouse gas emissions, yet one sector of anthropogenic pollution, light pollution is often overlooked. Awareness around this topic stems from education, policy and research. This issue which intersects ecological and social variables, is thus best studied under a systems approach, in which robust interdisciplinary perspectives are considered and analogously will warrant response from those in many fields. Contemporary studies of systems approaches frequently use agent-based models (ABM's) and techniques from network analysis. My project aims to pull from ideas in topology, ecology, and policy analysis to modify and strengthen light pollution policy in the contiguous United States.

2 Introduction

Systems theory, which describes the behavior and interactions of agents within a system, pulls from virtually all fields of academia. Coined by Australian biologist Ludwig von Beffalny, systems theory , "should be an important regulative device in science, to guard against superficial analogies that are useless in science and harmful in their practical consequences." It is the scientific movement against the advent of scientific reductionism and the call to view phenomena more holistically. Systems theory argues, as the Greek philosopher Aristotle puts, "The whole is greater than the sum of its parts". Under the field of general systems theory is a set of systems known as Complex Systems of Complex Adaptive Systems. Complexity, which within scholarship has become ambiguous, refers to a system which is composed of multiple and interconnected parts and looks to prediction instead of explanation. Complex systems are often difficult to describe and look to analytic techniques to understand them holistically. I defined light pollution as a complex system, as it is a problem of the intersection of ecological and social dynamics. Complex systems are ultimately identified by 4 characteristics, first, they exhibit self-organizing behavior, or organization ultimately emerges from complexity, second, they are non-linear, in other words a change to one component will have an effect on the system on a varying scale, huge, small or even none at all. Finally, they have a chaotic dynamic that ultimately becomes more ordered with time, and finally they show emergent behavior, or the collective and often unexpected behavior demonstrated by the mix of interacting components within a system. Emergence also figures its way into self-organization, in a bottom-up manner, where complexity emerges at the initiation of a system and gradually grows into nonlinear interactions between components or agents of a system, from here as a system grows increasingly complex, order surfaces. Some systems, and specifically the systems problem of Light Pollution are also adaptive, which means they change with perturbations, or disruption to the system to maintain an invariant or unchanging state, which in the case of light pollution would be the propagation of artificial light, by altering behaviors or structures, collectively known as properties within the system. Such properties of light would be its, duration, scattering, intensity and spectrum.

3 Background

3.1 ALAN

Artificial light at night or what I will consistently refer to by its acronym, ALAN, is the leading driver of light pollution and a component of a my system that describes light pollution. ALAN, plays a significant role in light pollution: as Dark Sky international reports, "Skyglow fouls the night sky for more than 80% of all people and more than 99% of the U.S. and European populations." Or as the authors on a recent study of light pollution summarize, "With an estimated annual average increase of 9.6% (estimated based on citizen science data), light pollution is one of the most pressing drivers of current global change, and it has become increasingly clear that the loss of the night has serious psychological, health, socio-economic and ecological consequences." ALAN is a form of "anthropogenic pollution", or pollution powered by human activity, that often disrupts the biological rhythms and in a systems view, the entire ecosystem. Humans, coincidentally are the prey of their own creation, as ALAN has detrimental effects on humans such as messing with circadian rhythms and sleep patterns.

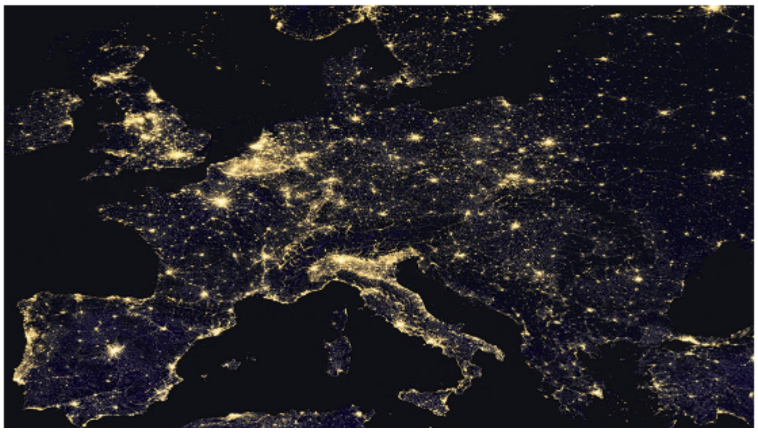


Figure 1: ALAN

3.2 Emergence and the Behaviors of ABM Ecological Systems

Emergent behavior is the emergence of order from complexity within a system and ultimately gives us insight into the dynamics of a system. Emergent behavior comes in many forms and is separated into two categories, strong emergence and weak emergence. Strong emergence, which came out of the movement of British emergentism in the 1920s and refers to a more philosophical understanding of emergence, that is not deducible. Weak emergence on the other hand, is the rise of high-level phenomenon from low-level domain that is deducible. It is thus this weak emergence phenomena that I am looking for. Weak-emergence often appears in the form of patterns, an example of which is in Conway's game of life, where particular patterns like the toad and glider emerge from the chaotic dynamics of Conway's game.

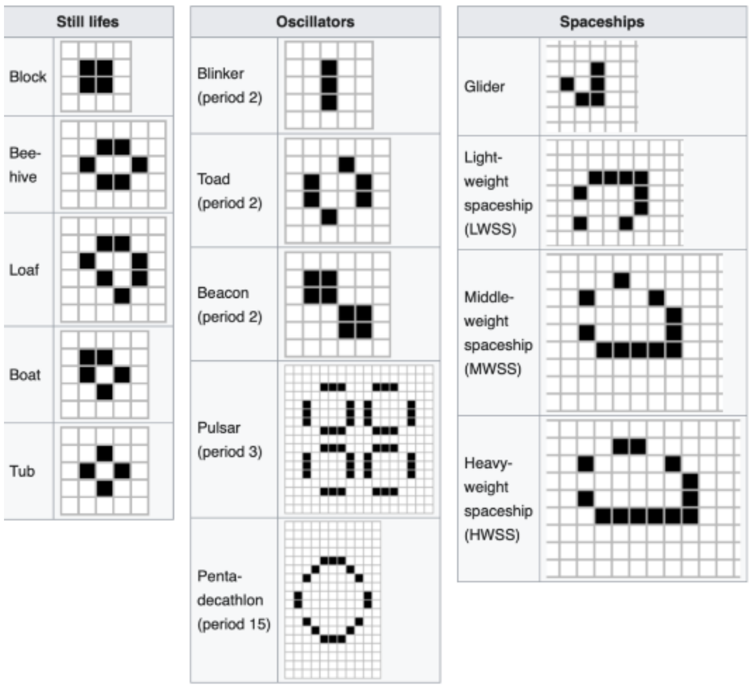


Figure 2: Conway's game of life emergent behaviors

My process of extrapolating these patterns and emergent behavior is similar in that it looks at patterns within the network representation of my model, by first analyzing the dynamics of the model. The underlying techniques are degree centrality, or the number of unique relationships or links a node has, betweenness centrality, or the measure of influence nodes have within a network, closeness, how connected a node is, eigenvector centrality, the discovery of central nodes, clustering tests, to cluster nodes into patterns, centrality tests, which once again identify central nodes, degree distribution or the number of connections per node, tests, modularity tests, or how well a network can be partitioned into modules, community detection algorithms, or the clustering of groups within the network and finally the number of connected components within a network.

3.3 Network Theory

Yet these metrics are virtually meaningless without proper introduction to the topological structures and properties of networks that make them so helpful in analyzing the emergent behavior of a system. Specifically, my model represents a socio-ecological network, or the connection of social(man-made) variables like light, and the ecological agents of the Orb-Web Spider ecosystem. Network Theory, which translates agents as nodes and interactions between agents as edges is advantageous because of its rich topological structure, from which patterns like modules, communities etc. reveal underlying behaviors and properties of a system. In the biological realm, Network Theory has seen growing applications in molecular biology as well as macro-scale ecological systems(in tandem with anthropogenic variables), like the one I sought to analyze. Ecological networks often characterize interactions between nodes as trophic or symbiotic, either a classical predator-prey relationship or coexistence respectively.

4 Model

In the vein of my systems approach, I created a system, an ecosystem in this case, resembling the real-life interactions between flora and fauna and this looming agent of light or ALAN. Now as I've highlighted before, because of both time constraints and processing power it was important to choose a system that illuminated the complexity of light pollution systems, whilst maintaining something feasible to model. Ok, so I've coined my model the "Orb-Web spider ecosystem" and it works like this: orb-web spiders, their insect prey, the prey's diet of grasses and finally lights are the agents within my model. Now, as recent studies have found, orb-web spiders are negatively impacted by the influx of night lighting, as they typically reside in the backyards, fences, walls and gardens that can support their intricate web designs. This intersection of ecosystem dynamics, emblematic in the orb-weaver spiders' natural lifestyle and social dynamics, the urban neighborhoods it often inhabits intrigued me as it would offer a level of complexity that paralleled the light pollution problem. Now as for the interactions between these agents, I limited them to growth and fecundity rates estimated for the spiders and their insect prey and the growth rate of the grasses, coupled with the basic food web dynamics of the system, or that the spiders eat the prey and the prey the grasses. With the addition of light, which were 18 in number given the average distribution of lights for an urban neighborhood, I added disruptor functions that changed the growth, fecundity among other properties of the insects and prey once within a certain diameter of the light. I visualized this model in a python library known as Mesa which is used for agent-based modeling and got the following results.

4.1 Agents

Pictured below is a snippet of the code in which I defined classes of spiders, insects, grasses and lights with their given properties and interactions between one another in a 2-d environment, much like the one in the late James Conway's game of life. In the code snippet, I define solely the spider agent.

```
class Spider(Agent):
def __init__(self, unique_id, model, age, fecundity, growth):
    super().__init__(unique_id, model)
    self.age = age
    self.satiation = 100
    self.fecundity = fecundity
    # self.grid = mesa.space.MultiGrid(width, height, True)
    self.growth_rate = growth
```

In the visualization, populations of lights, spiders, prey and grasses fluctuate. Dark green agents are spiders, yellow agents are of course lights, lime green agents are grasses and finally blue agents are the insect prey.

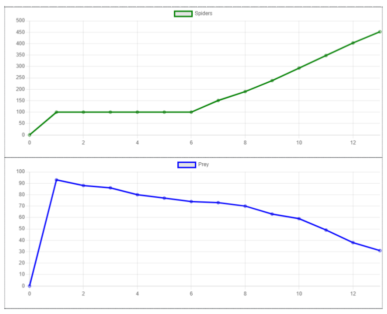


Figure 3: Spider-Insect population graph outputted by Mesa

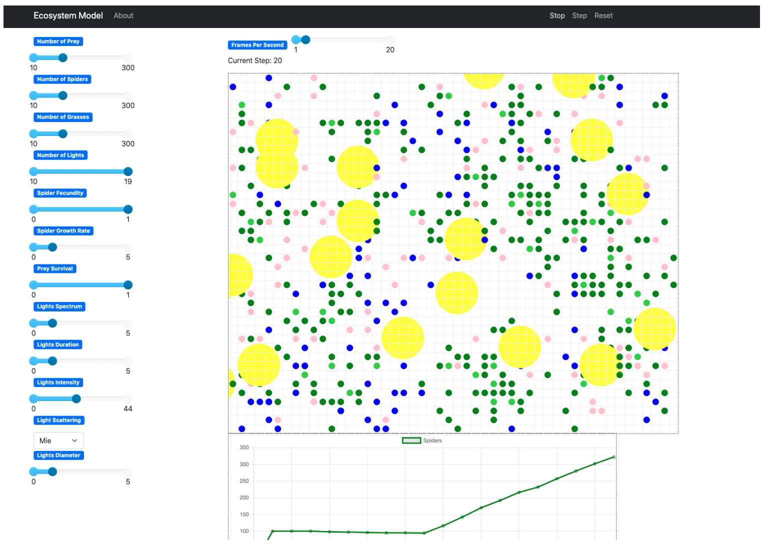


Figure 4: The Mesa Model GUI

Yet, to truly capture the properties and their relation to real-world phenomena, the aforementioned network theory held the answer. So how do we translate the Orb-web spider system into a network? The process is underscored by the topology of a network. Networks or graphs are defined as a set of nodes or vertices as they label them in this picture, connected by lines or edges. In a more holistic sense, the objects or agents within a system are nodes, and the links or the information that connects these nodes together are visualized as these links or edges. Now by exploring the topology of networks, we can uncover the underlying behaviors and properties of a system, so accordingly I translated my agent-based model into a network, in which the agents, like the spiders, grasses, lights and prey were the nodes and the edges were the interactions between them in their socio-ecological environment. I used the python library networkx to render this translation and ouputted a figure like the following, with randomly assigned values for parameters:

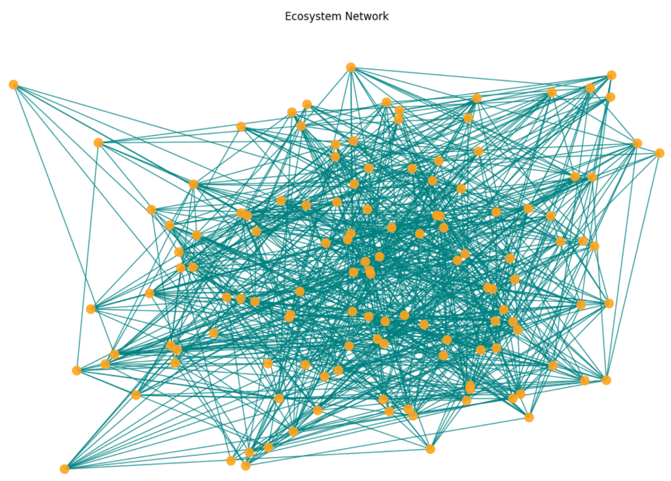


Figure 5: Network visualization of the ABM model

4.2 Verification

The verification of this model lies in observed interactions of spiders, light, insects and grasses across a multitude of sources and empirical data collected by researchers. However in the future I hope to create a more robust method of calibration and validation for my model. Yet, ultimately I was content in my decision to direct my focus towards the analysis of the model. After all, agent based models simply aren't capable of perfectly matching empirical data at the micro-level but rather responding to certain variables within that system most important to the modeller. Thus I employed face validation to verify my model against the conceptual 'Orb-Weaving Spider Ecosystem'. In tandem with face validation, however, there was one metric I used to test for statistical significance of the outputted data from my agent based models, or p-values of greater than 0.05. For my data I received the value of 0.0015689 which is less than 0.05 using the available functions under the SciPy library.

5 Methods and Results

In our given network, 4 parameters of the light agent/node are changed. These parameters are shown in table 1.

Table 1: Parameters

Parameter	**Description**
Spectrum	Spectral composition of light
Duration	The length of illumination periods during the nighttime
Intensity	How much light is there in lux/-candela?
Scattering	Reflected through aerosols and is ‘scattered’ throughout the sky

These parameters are defined by the following characteristics. Spectrum is an adjustable parameter, with a range of 320 nm to 1100 nm , since halogen lights which are typically used in city lighting are in this spectral range. Duration is simply a time spectrum parameter, where time (sec) is also adjustable. Intensity is another adjustable parameter that takes lux of light as an argument. Finally scattering is an adjustable parameter that gives the options of Rayleigh and Mie scattering. It occurs when the diameter of a particle interacting with photons is less than 50nm, or (where P is particle diameter) $P < \frac{1}{10}\lambda$. Mie scattering on the other hand occurs when interacting particles have diameters that occupy the range between 50-500nm, or follow:$\frac{1}{10}\lambda \leq P \leq \lambda$. Mie scattering of photons becomes more directional and is scattered forward, additionally, less light is scattered to the sides. Mie Scattering is described by the following equations: Q_s is proportional to:

$$x = (\frac{r}{\lambda})^4 \tag{1}$$

$$Q_s = \frac{\sigma_s}{\pi r^2} \tag{2}$$

Rayleigh scattering on the other hand follows the following equations:

$$Q_s = \frac{\sigma_s}{\pi r^2}$$

$$\left(\frac{r}{\lambda}\right)^4$$

Figure 6: Rayleigh Scattering

Where λ is wavelength and Q_s is proportional to

$$(\frac{r}{\lambda})^4$$

. Rayleigh scattering of photons on the other hand follows a peanut shaped distribution. The distributions of both are shown below:

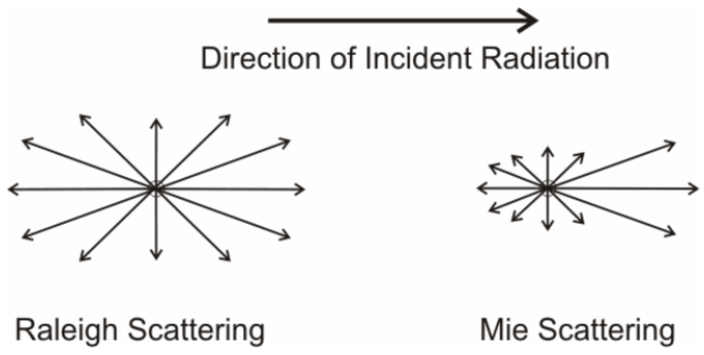


Figure 7: Rayleigh v Mie Scattering

I accordingly implemented this feature, in both the agent-based model and network representation of the system, where users can toggle between Mie and Rayleigh scattering in the system.

Thus adjustable parameters of the model include, spectrum, duration and intensity. These parameters thus act as independent variables, and by adjusting parameters allow us to view the behavior of the system under different light conditions, and thus offer recommendations and updates to current policy regarding urban light usage.

The question naturally becomes, how do we characterize behavior of our system?

The metric depends on three factors, resilience, stability and regime shifts. These metrics are the expected emergent behaviors of our given system. Through analyzing network topology and outputted data from the agent based model, such emergences are quantified and sufficiently characterize the positive and negative behavior of the model, which emerge under different parameter settings.

5.1 Resilience

I define system resilience as the property of a system to recover from perturbation or disruption. Similarly, if I target a system, how well will it hold from the attack. My definition emulates the one established in 1973, by CS Holling or that, "resilience is the ability of ecosystems to eliminate the impacts of external disturbances and maintain stability through their own repair." In biological and ecological networks, like the one I was analyzing, is often characterized by structural stability of the network. Thus I could analyze my network's topology to assess it's resilience. Thus, I implemented a edge attack algorithm, that visualizes the cascading and recovering impact of a perturbation or disruption to a connection between nodes or agents in the system. The algorithm works, by first selecting random edges to attack (delete) and subsequently deleting the edges with node degrees, or connections to other nodes in the graph,less than 2. The algorithm is depicted below:

Algorithm 1 Network attack algorithm

```
Require: :Remove Edge G
  while $N \neq 0$ do
      if $D \leq 2$ then
          remove edge
      else if $N == neighbor$ then
          remove edge
          add edge if it becomes isolated to N
      end if
  end while
```

After the algorithm terminates, I analyzed the topology of the network to assess its resilience. This was done by searching for the largest connected component in the network. Implemented through the built in networkx function, connected_components. Networks with larger connected components, or number of connected nodes in a partition of a network were valued as having higher resilience. These connected components would look like the figure below:

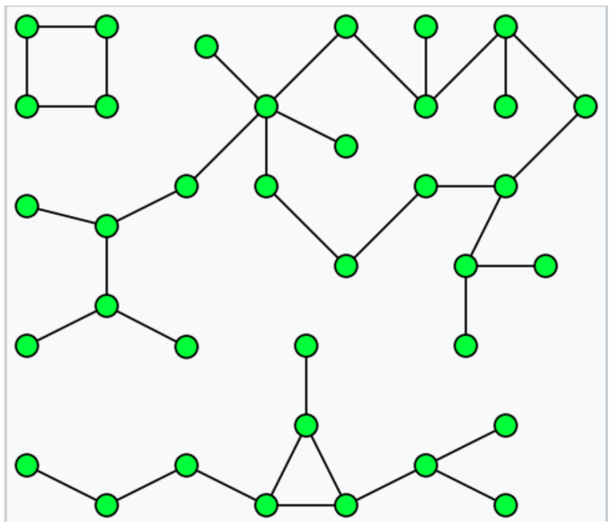


Figure 8: Connected Components

5.2 Stability

The structural stability of our ecological network is furthered by an additional factor, stability. Stability places structural resilience in a biological perspective, by studying the strength of an ecosystem in the face of adversity. The study of stability is rooted in zoologist, Charles Elton's stability-diversity hypothesis, that in ecological systems, biodiversity enhances stability of an ecological system. While recently met with revisions to consider the complexity of systems that contribute to stability, it is accepted that biodiversity is positively correlated with stability of an ecosystem. Thus, we can use biodiversity as an implicit metric for the stability of an ecosystem. Biodiversity is calculated through a metric known as Simpson's diversity index. The equation for the Simpson's diversity index is as follows: 1 minus the sum of the number of individuals within a group times the number of individuals minus 1 quantity over the total population times the total population minus 1. Or as summarized equation below:

$$D = 1 - \frac{\sum n(n-1)}{N(N-1)} \tag{3}$$

Biodiversity is calculated for every network iteration with adjusted parameters, by summing the number of grass nodes, spider nodes and prey nodes.

This biodiversity calculation is completed through a python function to calculate the Simpson diversity index. Biodiversity is then compared against topological features of the networks like modularity, eigenvector centrality, connectivity, closeness and betweenness centrality to test for correlation to biodiversity. This correlation was assessed by a metric known as one-way ANOVA, which measures for statistical difference between two groups.In addition, my process also used the f-statistic and p-values acquired from the variance analysis of ANOVA to measure statistical significance and correlation between two variables. This preliminary step is necessary, as their is ambiguity in scholarship around the general connection of network features and stability. Thus it was imperative that I ran my own assessments to determine the best topological metrics for stability All five of the topological features were easily implementable through the networkX library, as they were built in. Features that showed positive correlation (via finding Pearson correlation coefficients) to biodiversity were thus considered in assessing networks stability. Ultimately, the features that were chosen for highest correlation were used to assess the stability of the parametrically distinct systems. Those with higher values for a given topological feature would correlate to a higher level of stability. These features I identified were modularity, eigenvector centrality and connectivity.

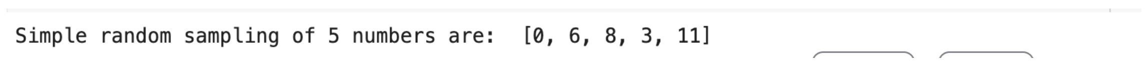
Additionally, another metric was or the presence of communities. In a similar vein to the biodiversity and topological feature approach, this metric saw perturbations in the form of distinct values of parameters for light, spectrum, intensity, duration and scattering. The presence of communities suggest higher stability within a system. The implementation of this approach, I scoured through popular community detection algorithms and decided upon the Girvan-Newmann algorithm, which removes edges to identify the tightly-knit communities underlying the network. More of these communities suggested more stability.

5.3 Regime Shifts

The final metric considered in my analysis, was the quantification of regime shifts. The idea of regime shifts is classical to most ecological research and describes the change in behavior of an ecological system, or the shift of ecosystems to different states. Catalysts to such transitions are called tipping points, or as I translated them in my analysis, change points. These change points anticipate the shift of ecosystem behavior over time and thus, I had to review the time series data outputted by my mesa model, where agent population was graphed against time, like the ones outputted in figure 3. I downloaded my data into a pandas dataframe and then into a csv, ready to be analyzed by a change point detection algorithm. My algorithm depended upon the ruptures python library. My algorithm implements a Reproducing Kernel Hilbert Space (RKHS) process that works by replotting my graphs to the high-dimensional vector space. My approach identifies change points through three available kernel functions in the rupture library, Linear, RBF and Cosine. This works as an optimization problem which segments the data based on the change points. At these change points, I used numerical differentiation to calculate the derivative at each of these change points. The proportion of negative to positive derivatives was calculated for each network.

6 Results

Both the ABM and network representations for my model were run 17 times,to explore the parameter space of the 4 independent variables I was looking at, spectrum, duration, intensity and scattering of light. To choose the sampled points of each parameter I employed simple random sampling to sample the parameter. For instance, to sample the duration parameter space I got a result like the following:



```
Simple random sampling of 5 numbers are:  [0, 6, 8, 3, 11]
```

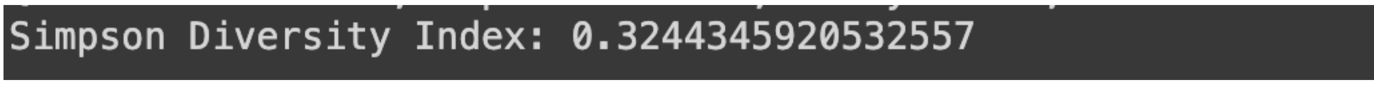
Figure 9: Output Sampling

6.1 Stability

I identified the features most closely correlated with biodiversity as eigenvector centrality, modularity, and closeness centrality of a network. Biodiversity was calculated through the following python function:

```python
import pandas as pd
data = pd.read_csv('data.csv')
species_counts = data.sum().to_dict()
print(species_counts)
def simpson_di(data):
    """ Calculate the Simpson Diversity Index from a dictionary of species counts.
    """
    N = sum(data.values())  # Total number of organisms
    if N == 0:
        return 0  # Prevent division by zero
    return sum((n / N) ** 2 for n in data.values() if n != 0)
diversity_index = simpson_di(species_counts)
# Print the result
print("Simpson Diversity Index:", diversity_index)
```

In which data.csv was the stored csv file of the output of my Mesa model's data collector for populations of spiders, grasses and prey. The python code outputted a result for a diversity index between 0 to 1. Which in the case of a run of my model to investigate the properties of a scenario in which lights had wavelengths of 707nm, or were approximately red in color I received the following result:

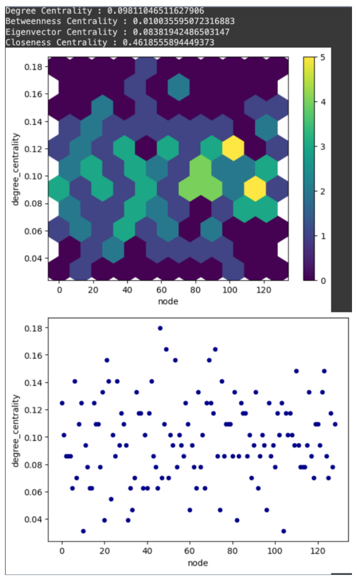


```
Simpson Diversity Index: 0.3244345920532557
```

Figure 10: Simpson's Diversity Index

To analyze the topological features of the network representation of my model, I simply implemented built-in functions of the NetworkX library, and plotted my results. In the case of a scenario in which photons were scattered via Rayleigh Scattering, I gathered the following outputs

The final correlated topological property of a network, modularity, or the density of connections within communities or modules of a network was written within a community detection algorithm. My algorithm graphed the modularity, calculated through a NetworkX function for each k-component or connected sub-graph within the network. For a source of ALAN that lasted 9 hours or epochs a day, the corresponding network produced the following modularity graph for each k-component:



```
Degree Centrality : 0.09811046511627906
Betweenness Centrality : 0.010035595072316883
Eigenvector Centrality : 0.08381942486503147
Closeness Centrality : 0.4618555894449373
```

Figure 11: Eigenvector Centrality

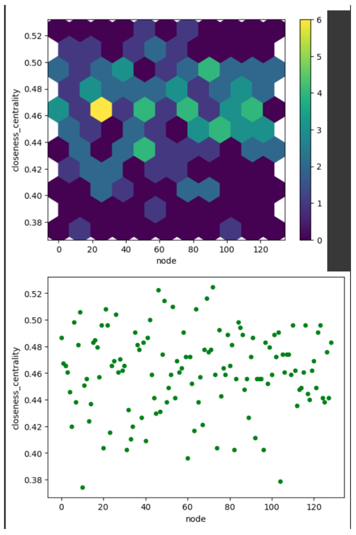


Figure 12: Closeness Centrality

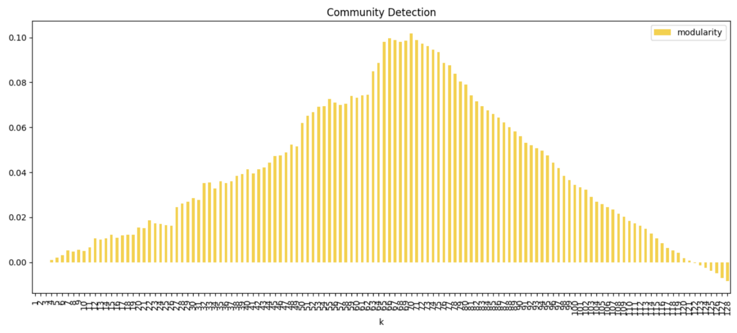
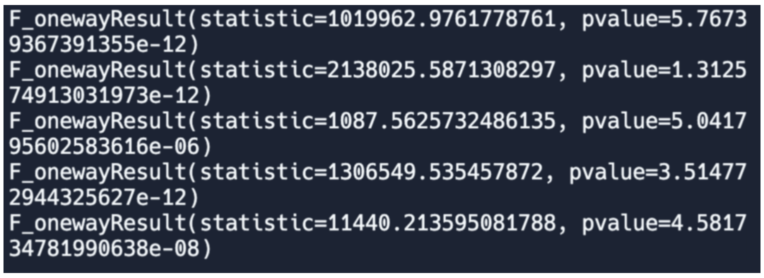


Figure 13: Modularity Graph

Such relevant topological features as proven by ANOVA f-statistics, calculated through available functions in python's SciPy library, shown below, like eigenvector centrality, modularity and closeness centrality were calculated for each set of sampled points within the four parameter spaces.



```
F_onewayResult(statistic=1019962.9761778761, pvalue=5.7673
9367391355e-12)
F_onewayResult(statistic=2138025.5871308297, pvalue=1.3125
74913031973e-12)
F_onewayResult(statistic=1087.5625732486135, pvalue=5.0417
95602583616e-06)
F_onewayResult(statistic=1306549.535457872, pvalue=3.51477
2944325627e-12)
F_onewayResult(statistic=11440.213595081788, pvalue=4.5817
34781990638e-08)
```

Figure 14: One-way ANOVA p-values and f-statistics for 5 topological features, Eigenvector Centrality, Modularity, Betweenness Centrality, Closeness and Degree Centrality

Since higher f-statistics indicate higher correlation between two variables, Eigenvector Centrality, Modularity and Closeness were chosen as the most indicative metrics of biodiversity and thus stability.

Some loose trends emerged, the aggregated eigenvector centrality of networks with variable light spectra, bore interesting results. For example, cooler lights or lights with wavelengths between 400 and 600, or violet to light green had the highest eigenvector centrality, of 0.8522 an 0.8546 respectively. Eigenvector centrality refers to nodes with the most influence over other nodes within their network, this may characterize the significant impact of warm colors on ecosystems. This was countered by a general trend, warmer lights contribute to greater stability within an ecosystem, as the aggregated score of modularity, eigenvector centrality and closeness centrality was higher with warmer colors between the spectrum of 650 to 750nm or deep orange to red, with aggregated scores almost 0.2 greater than their cooler color counterparts. Furthermore, 3 hours of ALAN usage proved the most network stabilizing, as it consistently ranked highest through the aforementioned topological features, and earned an aggregated score of 0.1894116, the highest of any of the changed variables for duration. For intensity, 20 lux proved the most stabilizing with an aggregated score of 0.4343086667, the highest of any of the changed variables. In scattering, Mie scattering was more stabilizing with an aggregated score of 0.1822181333. This outcome is consistent with our other findings, since Mie scattering often prefers or occurs with photons of longer wavelengths, or warmer colors on the light spectrum.

Additionally, other metrics were used to analyze the stability of our given network. The process was initialize through built-in functions of NetworkX like community detection algorithms. To commence my analysis I use one such built-in function, that identified communities through the Girvan- Newmann algorithm. The code for the aforementioned steps is below:

```
G = model.G
communities = list(nx.community.girvan_newman(G))
print(communities)
```

This code in tandem with visualization libraries and the discovery of modularity for individual nodes or agents in a system resulted in results like the following (specifically for 707nm wavelength of light):

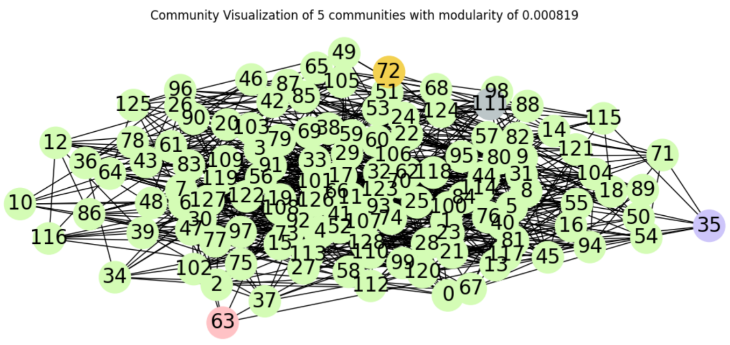

Community Visualization of 5 communities with modularity of 0.000819

Figure 15: Communities Graph, note colors signify different communities. In this image there are 5 colors corresponding to 5 identified communities

My code also allowed me to look at the largest connected component in the network of the socio-ecological system, as well as the degree, or how many edges branched off from a node in the network in the images below:

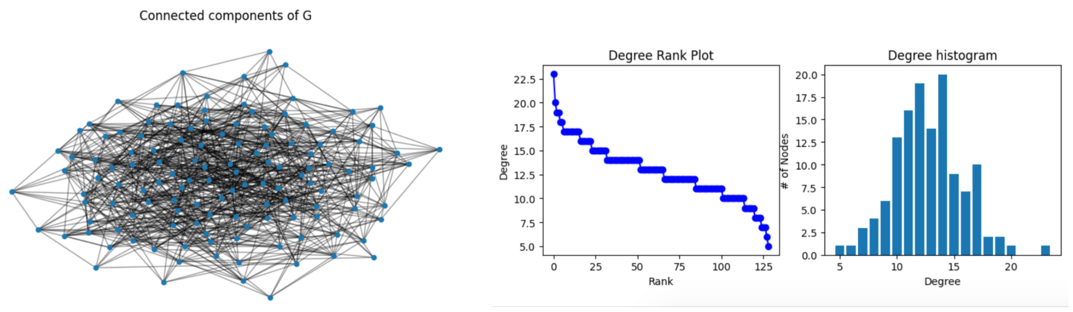


Figure 16: Topological structure

Both degree centrality and connected components are good metrics of stability, because they reveal the graphs and by relation, the system's ability to tend towards becoming a k-connected graph, in which removing edges from the network never disconnects the network.

Communities were most prevalent at certain conditions within the investigated parameter space. For example, scenarios with high modularity resulted in a greater number of communities. This logic is consistent, as modularity measures the degree to which a network can be partitioned or separated. This logic follows S.Gomez and P.Jensen's Balance Theory which states the following, where Q_E^- and Q_E^+ refer to the negative and positive edges that may separate communities. Negative edges form out of severed relationships, like the prey and the spider, and ultimately particular settings of light and the organisms within the system. Positive edges on the other hand form from positive relationships between organisms in this case:

$$Q_E = \frac{2\omega^+}{2\omega^+ + 2\omega^-} Q_E^+ + \frac{2\omega^-}{2\omega^+ + 2\omega^-} Q_E^-,$$

$$Q_E^+ = \frac{1}{2\omega^+} \sum_i \sum_{j(i \neq j)} \left(\omega_{ij}^+ - \frac{\omega_i^+ \omega_j^+}{2\omega^+} \right) \delta\left(C_i, C_j\right),$$

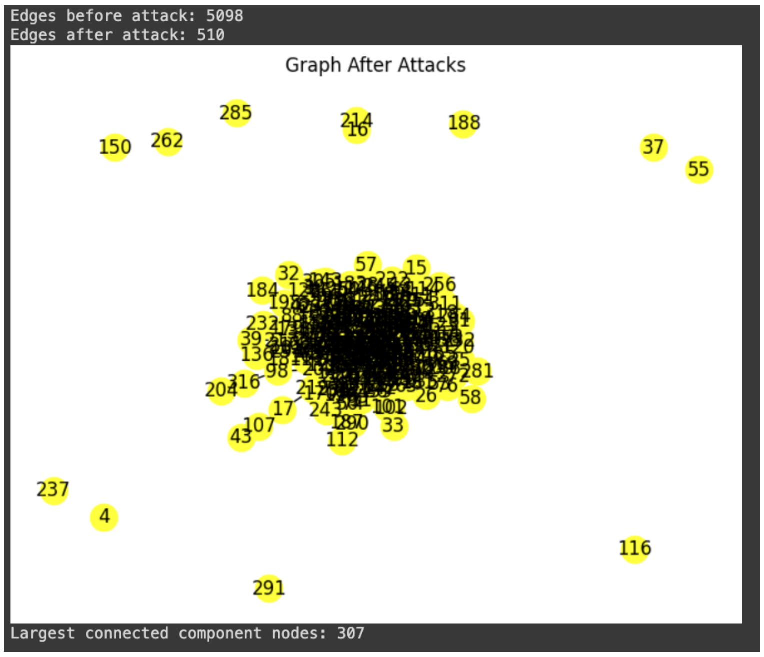
$$Q_E^- = \frac{1}{2\omega^-} \sum_i \sum_{j(i \neq j)} \left(\omega_{ij}^- - \frac{\omega_i^- \omega_j^-}{2\omega^-} \right) \left(1 - \delta\left(C_i, C_j\right)\right),$$

Figure 17: Balance Theory

For spectrum the highest number of communities was discovered for lights with wavelengths of 686 nm. For intensity, the setting of the lights with 20 lux produced the greatest number of communities, for duration 3 hrs of ALAN usage once again prevailed amongst the other parameter settings and held the highest number of communities. Mie scattering triumphed over Rayleigh scattering for the greatest number of communities.

6.2 Resilience

The analysis and emergence of stability in a network offered insight into the properties of the ecosystem, by associating topological features with observed properties of the system, like biodiversity or population statistics. Resilience, conversely, quantifies properties of the representative network of an ecosystem, yet still lends much insight into the structural robustness of our system. Thus I employed the previously defined node attack algorithm to measure the structural resilience of a network. Using a python algorithm to attack nodes, for a scenario in which the intensity of lights was 33 lux, I uncovered the following result or output from my algorithm. The largest connected component was subsequently outputted afterwards.

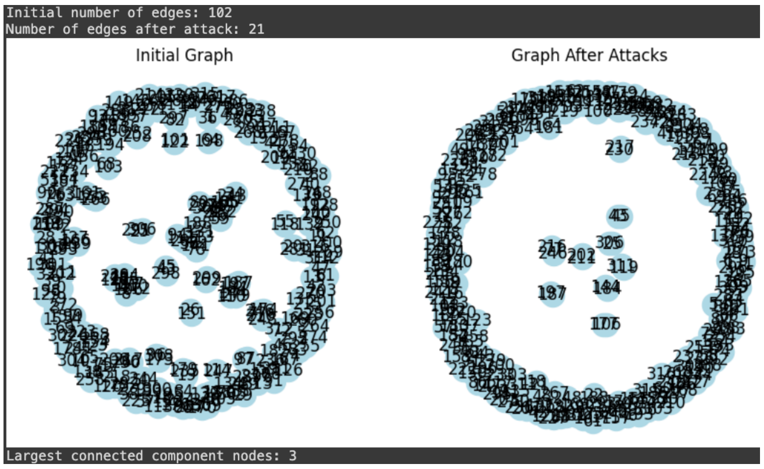


```
Edges before attack: 5098
Edges after attack: 510
```

```
Largest connected component nodes: 307
```

Figure 18: Graph after node attack

Additionally I ran the algorithm multiple times as an additional metric of measuring how long a network could withstand the node attack algorithm. The second and third runs of the node attack algorithm on the network with light intensities of 33 lux are shown below:



```
Initial number of edges: 102
Number of edges after attack: 21
```

```
Largest connected component nodes: 3
```

Figure 19: Graph after multiple node attacks

Resilience was ultimately assessed by the number of connected components, still intact after the node attack algorithm had affected a network once. Unfortunately there weren't any clear patterns that emerged from the largest connected component analysis, which opens up further questions on the disjointed relationship of structural resilience to outwardly expressed properties of systems like stability and regime shifts, or a slight error in my code that might shield such results. As shown in the graph below, the values for largest connected component for each value of stayed level across the parameter range of light intensity, and bears only a slight resemblance to a parabolic curve.:

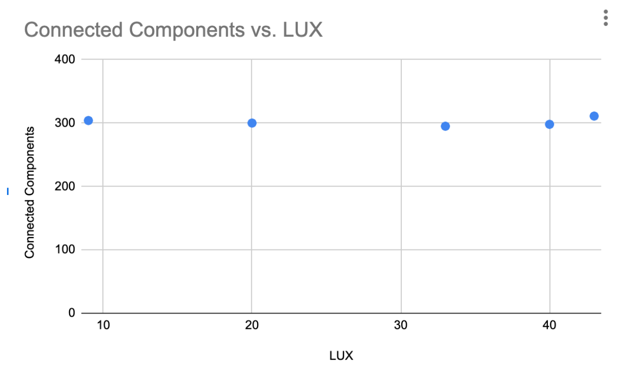
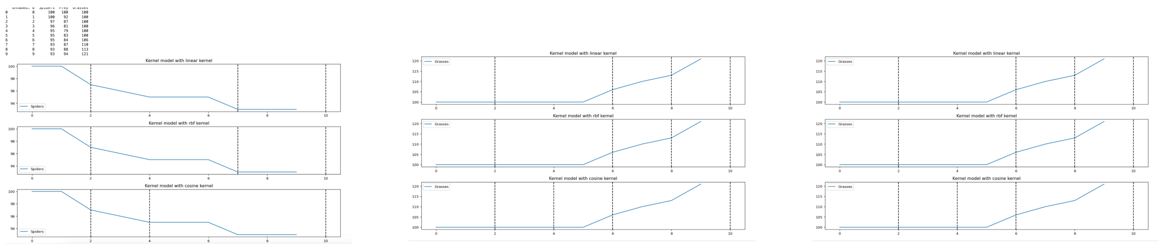


Figure 20: Exceptions

Similar relationships were identified for the rest of the three independent variables.

6.3 Regime Shifts

Regime Shifts, the final component to my analysis, partition the data outputted by the Agent-base model by change points, or slight shifts in the state and behavior of populations of, in this case, spiders, prey and grasses. I used three kernels and ultimately identified the cosine kernel as the overwhelmingly worst kernel to identify such change points. The Linear and RBF kernels were much better at doing so, as exemplified by the change point detection algorithm output on a scenario with lights wavelengths of 414nm:



```
   Unnamed: 0  Spiders  Prey  Grasses
0           0      100   100      100
1           1      100    92      100
2           2       97    87      100
3           3       96    81      100
4           4       95    79      100
5           5       95    83      100
6           6       95    84      106
7           7       93    87      110
8           8       93    88      113
9           9       93    94      121
```

Figure 21: Kernels change point detection for spiders, prey and grasses

Regime shifts, after identified through change point detection were thus classified as either positive or negative change, by simply calculating the derivative of the change point through a simple python implemented numerical differentiation algorithm below which takes a dictionary of points uploaded from my data.csv file:

```
    h = x[1] - x[0]  # Increment between adjacent x values
derivative_at_index = (y[index + 1] - y[index]) / h
print("The derivative at index", index, "is approximately:", derivative_at_index)
if derivative_at_index > 0:
  print("positive change")
else:
  print("negative change")
```

The results approximately followed those observed in the stability section, with the most positive change at a wavelength of 686 nm for the spectrum parameter with an overall derivative of 1.357, at 9 lux for the light intensity parameter with an overall derivative of 1.245, at 3 hours for the duration parameter with an overall derivative of 1.578 and finally Mie scattering prevailed once again over Rayleigh scattering with an overall derivative of 1.286. Additionally, as I outputted 3 kernel functions to approximate the change points, on average the RBF and Linear kernel functions approximated the change points more accurately, but there were some exceptions. For example, for the scenario in which ALAN lights had a wavelength of 591 nm, the Cosine kernel function proved one of the better approximations:

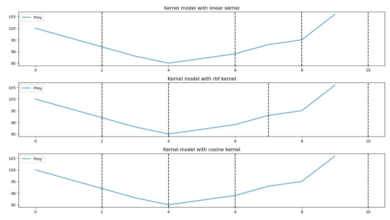


Figure 22: Cosine Kernel Function for Prey

7 Policy

7.1 Existing Policy

Policy on ALAN is quite variable throughout the contiguous United States, and becomes even more pronounced throughout the world. Thus, for the scope of my project, I analyzed the current in place policies in New Mexico, USA.

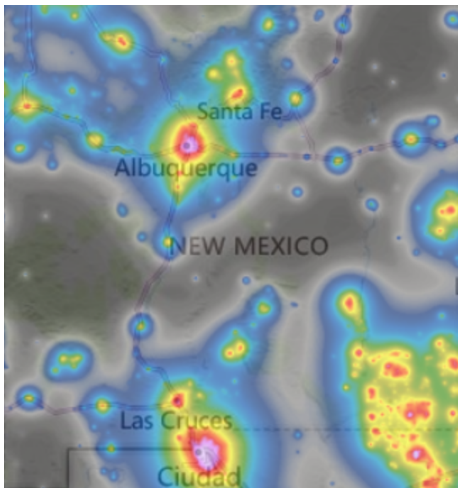
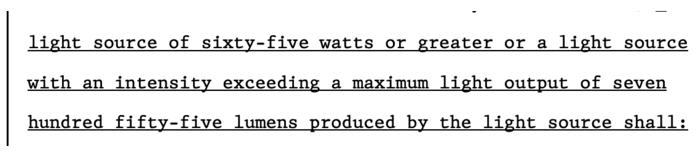


Figure 23: New Mexico ALAN map

The Dark Skies Conservation Trust of Santa Fe instituted the Night Sky Protection Act in 1999, which, a novelty for it's time, "makes dark skies a priority in New Mexico for the health of its people, wildlife, and economy. The act requires that outdoor lighting be fitted with shielding that directs light downward, rather than upward or laterally." This light positioning ensures the view of a partially or fully visible night sky, however its effects on biological systems have not been thoroughly investigated and advised. As Ed Yong, a science writer for the Atlantic summarizes in his novel, *An Immense World*, "The daily and seasonal rhythms of bright and dark remained inviolate throughout all of revolutionary time–a 4-billion-year streak that has began to falter in the 19th century." Artificial lighting thus proves a threat to the 'sensory-scapes' of a great variety of species, in addition to obscuring our view of the night sky. Position of lights is thus not as important to the condition of an ecosystem, but rather properties of light like spectrum,duration,intensity and scattering are more crucial to consider. In the New Mexico House Bill 461 of 2009, 10 years after the Dark Skies Conservation Trust instituted their efforts, made slight revisions to the outdoor lighting policies in New Mexico. Most notably was a sole mention of intensity of light provisions as stated below:



light source of sixty-five watts or greater or a light source with an intensity exceeding a maximum light output of seven hundred fifty-five lumens produced by the light source shall:

Figure 24: Bill 461

Thus my recommendations lie in reassessing this statement as well as adding potential updates to the house bill based on my findings above.

7.2 Recommendations

Note: While I have results, before the expo, I want to ensure the accuracy of my data before adapting the House Bill 461 to better suit my findings, however I know that adding provisions to change to warmer colors and accordingly, larger wavelengths of light is imperative, as well as decreasing duration to around 3 to 6 hrs a day of light, and keeping intensity at around 9 to 30 lux is preferable for the health and success of surrounding ecosystems and natural habitats.

8 Conclusion

The central question of my research calls for awareness around light pollution. The answer lies in policy. Policy and law after all is our best measure to enforce rules and influence the public. As Steve Kaisler at the university of Notre Dame explains, "The nature of complex systems can be assessed by investigating how changes in one part affect the others, and the behavior of the whole." Thus by making minor changes to the light agent, I hope to explore and compare the emergent behavior of these different networks to inform infrastructural and social policy. Ultimately at the heart of my process is the use of science for the embetterment of the world, this same sentiment is what drives others in the scientific field, and the other systems of people and organisms in the world. It is the emergent behavior to do good that unites us, and to recognize the unsung systems around us.

9 Achievements

My greatest achievement was learning how to use both the NetworkX python library and the MESA agent-based modeling library as well as exploring systems and network analysis and how they could be interpreted in a biological framework.

10 Acknowledgments

I'd like to thank my sponsoring teachers Ms.Jocelyne Comstock, Ms.Gabriella Masoni for providing the space for me to work and help from Dr.Mark Galassi on this project.

References

[1] Cong, Wei, et al. "The Coexistence Relationship between Plants and Soil Bacteria Based on Interdomain Ecological Network Analysis." Frontiers, Frontiers, 8 Nov. 2021, www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2021.745582/fullsupplementary-material.

[2] He, Xiaochen, et al. "A generalized modularity for computing community structure in fully signed networks." Complexity, vol. 2023, 24 Feb. 2023, pp. 1–20, https://doi.org/10.1155/2023/8767131.

[3] e-Yu, Chao. "ANOVA Test, with Python." Medium, Towards Data Science, 5 Jan. 2023, towardsdatascience.com/anova-test-with-python-cfbf4013328b.

[4] olling, C. S. "Resilience and stability of Ecological Systems (1973)." The Future of Nature, 31 Dec. 2017, pp. 245–260, https://doi.org/10.12987/9780300188479-023.

[5] Yong, Ed. An Immense World: How Animal Senses Reveal the Hidden Realms around Us. Random House, 2022.

[6] Farrell, Damien. “A Simple Agent Based Infection Model with Mesa and Bokeh.” Bioinformatics and Other Bits - A Simple Agent Based Infection Model with Mesa and Bokeh, dmnfarrell.github.io/bioinformatics/abm-mesa-python. Accessed 18 Apr. 2024.

[7] Bastolla, Ugo, et al. “The architecture of mutualistic networks minimizes competition and increases biodiversity.” Nature, vol. 458, no. 7241, Apr. 2009, pp. 1018–1020, https://doi.org/10.1038/nature07950.

[8] Nature News, Nature Publishing Group, www.nature.com/scitable/knowledge/library/biodiversity-and-ecosystem-stability-17059965/. Accessed 18 Apr. 2024.

A Realtime Camera Fusion 3D Model with a Novel Feature-Matching and Star Identification-Based Calibration for Tracking Smoke Plumes

New Mexico

Supercomputing Challenge

Final Report

April 9th, 2024

V. Sue Cleveland High School

Team Members:

Gene Huntley

Teacher(s):

Ashli Knoell, Leah Felty

Project Mentor

Stephen Guerin

Contents

Executive Summary

Smoke plumes produced by wildfires pose a threat to public safety. However, a lack of accurate real-time information on smoke plumes provided by current methods like satellite imagery leaves communities unable to make informed decisions critical to ensuring the safety of themselves and their families. Despite the current lack of smoke plume information from satellites, there is a lot of photography of smoke plumes. This project aims to provide an accurate and quick method to locate smoke plumes through smartphone cameras to utilize this untapped source of information. The research aims to find a method to calibrate a camera using manual and automatic selections of similar or known locations, such as celestial objects. After calibration, the proposed method renders a real-time 3D smoke plume with planar-to-planar correspondences. This algorithm proves to be significantly faster than current satellite imagery methods (MODIS and HMS) while providing an improved level of granularity than current surveillance methods.

Introduction

Problem

An issue facing urban and rural populations is a lack of reliable smoke plume surveillance, leaving communities and local authorities with no information on dangerous smoke that can harm them. Specifically, climate change has increased global temperatures, drastically increasing the length and duration of wildfire seasons (McKENZIE et al., 2004). As a result of an increase in the frequency of wildfires, there has also been a significant increase in the number of smoke plumes emitted by said wildfires. This increase can pose a risk to public safety due to the emission of toxic particles such as PM2.5 by smoke plumes, causing chronic illnesses such as asthma and acute heart attacks (Haikerwal et al., 2015; Dennekamp and Abramson, 2011). Furthermore, a lack of information on smoke plumes can harm federal authorities conducting prescribed burns. Without information on the location and spread of smoke plumes, prescribed burns - necessary for understanding wildfires and clearing vegetation buildups - can threaten public health (Wu et al., 2023). Smoke plumes not only have the potential to cause death from poor air quality, but it also affects a large number of people; in the United States alone, 1.6 million people live in areas with dangerous air quality caused primarily by smoke plumes (Kelishadi and Poursafa, 2010). Despite the dangers smoke plumes can pose to public health, there is a severe lack of reliable real-time smoke plume information. Thus, the research aims to create a network of phone cameras to render a smoke plume and provide real-time surveillance information on the aforementioned smoke plumes.

The most widely used method to gather information on smoke plumes is through remote sensing, a process that collects visual information on orbital bodies like Earth (Earth Science Data Systems, 2019). One of the most extensive smoke plume algorithms that analyzes remote sensor information for smoke plumes is the National Oceanic and Atmospheric Administration's (NOAA) Hazard Mapping System (HMS) (Data and Service, 2019). In addition, many smoke plume monitoring systems also rely on NASA's Moderate Resolution Imaging Spectroradiometers (MODIS) and Visible Infrared Imaging Radiometer Suite (VIIRS) mounted on various satellites (NASA, 2000; Earth Science Data

Systems, 2019; Xie et al., 2007). However, a barrier that prevents organizations from utilizing remote sensing to monitor the spread of smoke plumes is the inability of remote sensors to capture images with sufficient resolution and speed. According to NASA, the resolution of MODIS is 1km, while the resolution of NOAA's HMS is 5km (Earth Science Data Systems, 2019; Data and Service, 2019). In addition, the time granularity of the remote sensors - the time between each capture - is a day, causing tracking small to medium smoke plumes live with remote sensing to be exceedingly challenging (Earth Science Data Systems, 2019).

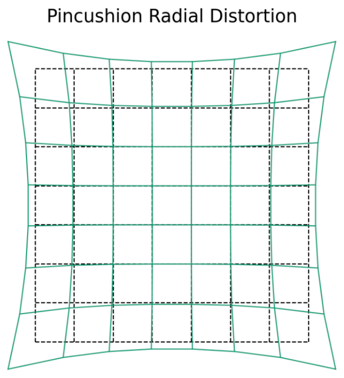
Currently, 93% of the global population owns cell phones, most of which are capable of video and image recording (Center, 2022). Accompanied by the massive increases in the processing capabilities of cell phones, smartphones can collect and analyze necessary data for analysis (Muhammad et al., 2018). Therefore, instead of relying on outdated forms of surveillance that require satellites, this research aims to construct a network of phones and other cameras to monitor the spread of smoke plumes (Almalkawi et al., 2010).

The proposed network calibrates each of its cameras, utilizing information ranging from celestial positions to feature identification to obtain the position, rotation, and distortion parameters of the camera. Finally, the algorithm uses visual media from various calibrated cameras to visualize a 3D point cloud or mesh of a smoke cloud, giving information on smoke plume geometry and location. Ideally, the user should be able to take a calibration and rendering image to produce a smoke plume. By relying on community-sourced information, the proposed method overcomes the lack of spatial and temporal granularity provided by remote sensing devices due to its speed and accuracy when creating 3D smoke plumes.

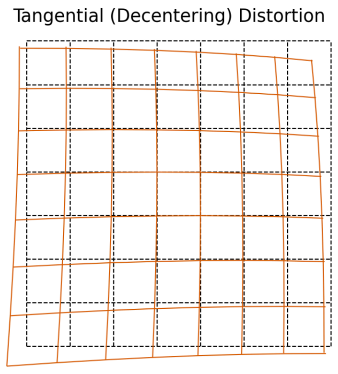
Background

Camera Model

A camera has both radial and tangential distortions, which are distortions that result in a misalignment of an image, as seen in Figure 1. Since lens distortion can cause triangulation errors over large distances, each pixel in the image was adjusted using a 4-parameter Brown-Conrady distortion model, a model that fixes the image distortion given a few distortion parameters (Brown, 1966).



(a) Radial Distortion



(b) Tangential Distortion

Figure 1: Distortion Examples, Source: Steward, Jeremy. "Symmetric Distortions." Camera Modeling: Exploring Distortion and Distortion Models, Part I, TangramVision, 6 Aug. 2021, www.tangramvision.com/blog/camera-modeling-exploring-distortion-and-distortion-models-part-i.

Examples of image plane offset as a result of distortion.

Every pixel on an image plane has a position relative to the rest of the image defined as (u, v), which a projection matrix converts into 3D coordinates given known camera rotation and position parameters (OpenCV, 2021). Given only the (u, v) pixel coordinates are known, the rest of the paper will focus on solving for the unknown variables presented in the camera model needed to render a smoke plume. A camera with arbitrarily set rotation, distortion, and translation parameters and a fixed depth for each vertex can be seen in Figure 2.

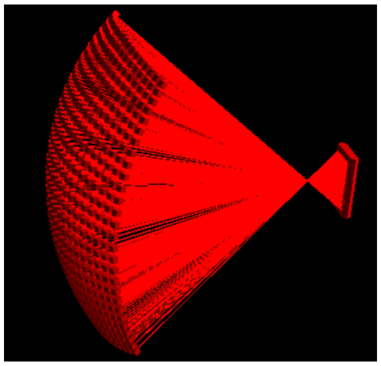


Figure 2: Simulated Camera

A visualization of a camera with each pixel projected at a fixed depth.

Levenberg-Marquardt and ECEF Coordinates

Levenberg-Marquardt algorithm, a popular variation of gradient descent commonly used to approximate the local minimum of a function, will be a heavily relied-upon algorithm (Gavin, 2022; Moré, 1978). Each iteration of the Levenberg-Marquardt increases each parameter by β_n, where w_n is a parameter such as position or tilt, and λ is a constant that is changed during each step of the algorithm:

$$J = \begin{bmatrix} \frac{\delta w_1}{\delta E_1(x)} & \cdots & \frac{\delta w_1}{\delta E_n(x)} \\ \ldots & \ldots & \ldots \\ \frac{\delta w_m}{\delta E_1(x)} & \cdots & \frac{\delta w_m}{\delta E_n(x)} \end{bmatrix} \tag{1}$$

$$\beta = (J^T J + (\text{diag}(J^T J)\lambda))^{-1}(E_n(x)J^T) \tag{2}$$

To calibrate a camera, ideally, the research would need an error function, $E(x)$, as described in the above equation, such that the perfect calibration of a camera results in the minimum of said error function.

The Earth-Centered-Earth-Fixed coordinate system, a coordinate system that uses the center of the Earth as the origin is used heavily (Zhu, 1994). Although a latitude-longitude-elevation coordinate system produces negligible error at short distances, not accounting for the Earth's curvature at long distances can result in errors. To convert local camera coordinates into ECEF coordinates, the below change-in-basis is used, where $< X, Y, Z >$ is the ECEF coordinate of the focal point, and $< x, y, z >$ represents the local coordinate of a pixel (Smith, 2022).

$$\vec{a} =< X, Y, Z > \tag{3}$$

$$\vec{b} = \vec{a} \cdot < 0, 0, -1 > \tag{4}$$

$$\vec{c} = \vec{a} \cdot \vec{b} \tag{5}$$

$$f(x, y, z) = \vec{a}x + \vec{b}y + \vec{c}z \tag{6}$$

Hypothesis/Research Goal

This research aims to use cellphones and other devices capable of multimedia recording to create a 3D representation of a smoke plume, utilizing a multiple methods for convenient calibration and deployment of cameras.

Methodology

Calibration

Before any operations can extract information on a smoke plume's location, a camera's distortion, translation, and rotation parameters need to be known.

Star Identification

One method to obtain $E(x)$ would be to use an error in the projected locations of stars in the night sky and their respective celestial positions. The proposed calibration algorithm employs a baseplate solver - an algorithm that identifies stars in an image - to identify the locations of the stars in the night sky, as shown in Figure 3.

After the algorithm identifies the (u, v) positions and names of certain stars in an image, it references the star's current right ascension and declination (the star's relative position to the Earth) from Harvard's Astrophysics Data System for information. Afterward, the algorithm uses a projection matrix with arbitrarily set camera parameters to convert the stars' (u, v) coordinates into ECEF coordinates. Levenberg-Marquardt then attempts to

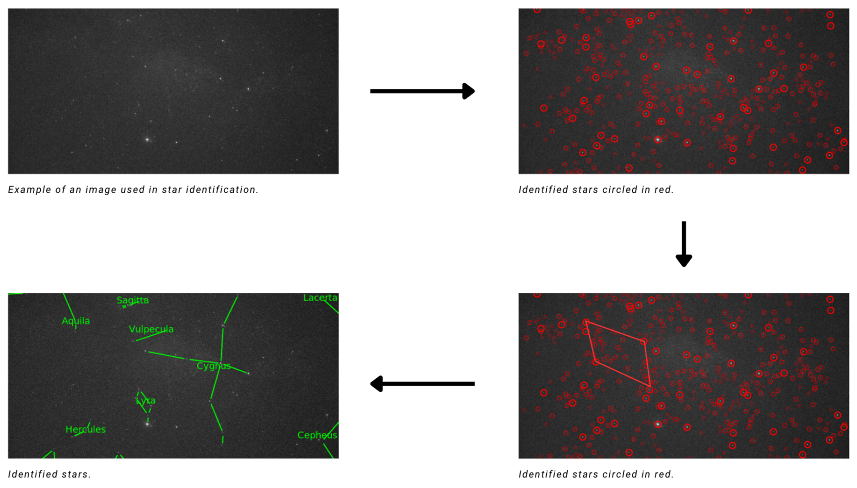

Example of an image used in star identification.
Identified stars circled in red.
Sagitta
Lacerta
Aquila
Vulpecula
Cygnus
Lyra
Hercules
Cepheus
Identified stars.
Identified stars circled in red.

Figure 3: Baseplate Solver

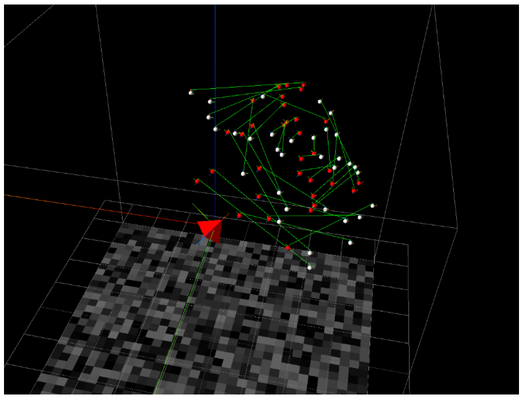
A brief description of how a baseplate solver identifies stars.

find the local minimum of $E(x)$, where $E(x)$ is defined as the squared altitudinal and azimuthal difference between the projected and actual positions of the stars. Figure 4 shows the Levenberg-Marquardt algorithm in use to reduce the error between the altitude/azimuth of the projected stars and their actual positions.

Point Correspondence Calibration

For situations where the stars in the night sky aren't visible, another powerful method of calibration is through point matching. In essence, a user selects two corresponding points, one in an image plane and another in a separate image plane or the real world.

The former method, the image-to-image point correspondence, is described as follows: within two arbitrary images, matching pixels are selected automatically or through user input. Those pixels are projected with the projection matrix into the real world at an infi-


(a) Projected Stars before Optimization

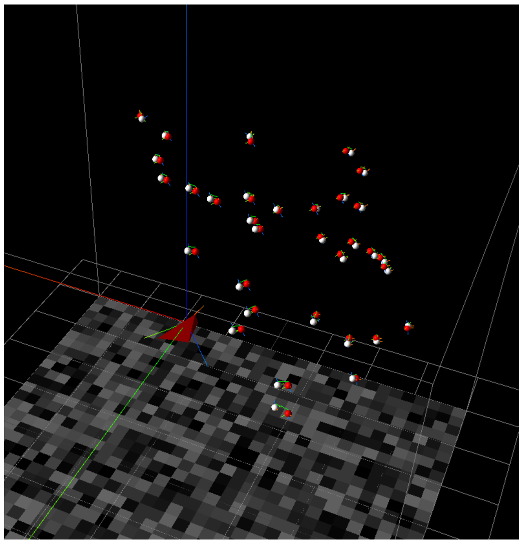

(b) Solved Camera Parameters

Figure 4: Solved Stars

An example of a camera with calibrated and uncalibrated parameters.

nite radius. After the two projected pixels are created for each correspondence, the error function is defined as follows, where (x, y, z) describes the position of the pixel on the image plane as a function, and (a, b, c) describes the position of the focal point:

$$r_n^2 = x_n^2 + y_n^2 + z_n^2 \tag{7}$$

$$q = x_1 x_2 + y_1 y_2 + z_1 z_2 \tag{8}$$

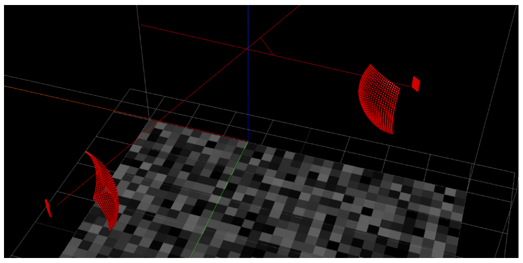
$$\Delta n = n_2 - n_1 \tag{9}$$

$$\begin{bmatrix} t_o \\ s_o \end{bmatrix} = \begin{bmatrix} r_1^2 & -q \\ q & -r_2^2 \end{bmatrix}^{-1} \begin{bmatrix} \Delta a x_1 + \Delta b y_1 + \Delta c z_1 \\ \Delta a x_2 + \Delta b y_2 + \Delta c z_2 \end{bmatrix} \tag{10}$$

$$f(x, a) = (s_o x_2 + a_2 - t_o x_1 + a_1)^2 \tag{11}$$

$$E(s, t) = \sqrt{f(x, a) + f(y, b) + f(z, c)} \tag{12}$$

The model of the camera function with the calculated errors represented as a red line between the pixel projections is shown in Figure 5.


(a) Projected Points and Error

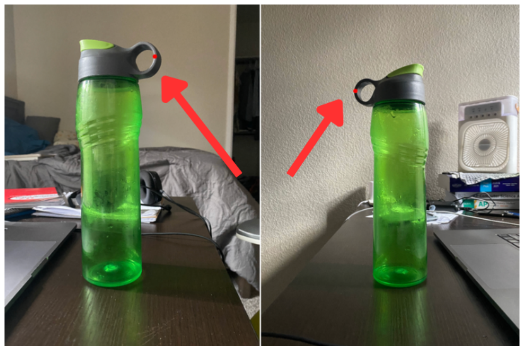

(b) Selected Points

Figure 5: Multiple-Image Calibration

An example of how multiple images can be used to calibrate a camera.

Note that if the camera is perfectly calibrated, the error function will be at its absolute minimum. After all, for a perfectly positioned camera in the network, two lines drawn by the projected pixels should intersect at the pixel's position. Thus, Levenberg-Marquardt is run on the error function to calibrate a camera.

A similar method to using two points on separate image planes is to utilize a correspondence between a real-world feature and an image correspondence. The method is similar to the previously described image-to-image correspondence. However, instead of projecting two lines from two cameras, one line is projected from the camera to the image plane, while another is projected from the center of the Earth to the selected point. An example is shown in Figure 6.

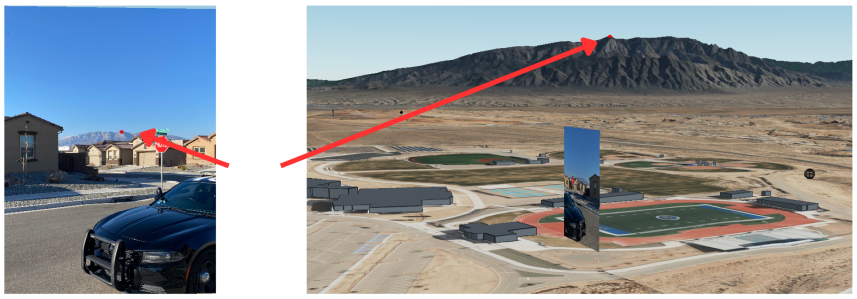


Figure 6: Simulated Camera

An example of how a single match between an image and a real-world correspondence can calibrate a camera.

Rendering

With all the necessary information about a camera, such as its position, rotation, and distortion, an algorithm can extrapolate 3D information from an image's data.

LoFTR and Manual Pixel Matching Tracking with the Lukas-Kanade Method

Before any image matching can be done, Meta's Segment-Anything-Model segments all the objects in an image (Kirillov et al., 2023). After segmentation, the user should select the approximate location of the smoke plume to crop the mask from the rest of the image, as shown in Figure 7. To plot out pixel matches and render a smoke plume, the researcher then applies LoFTR, an algorithm that extracts and matches features in an image, to find pixel matches in smoke plumes (Sun et al., 2021). An example of LoFTR in action is seen in Figure 8. Given the homogeneity of clouds and smoke plumes, there are bound to be many false positives. Therefore, a pixel match is only accepted if the algorithm is more than 50% confident of its accuracy.

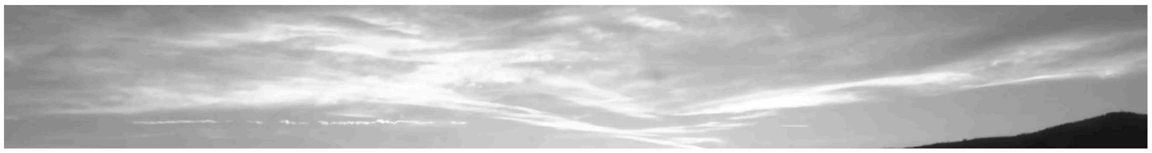


Figure 7: Cloud Cropped with SAM

A cropped cloud manually selected by a user after image segmentation by SAM.

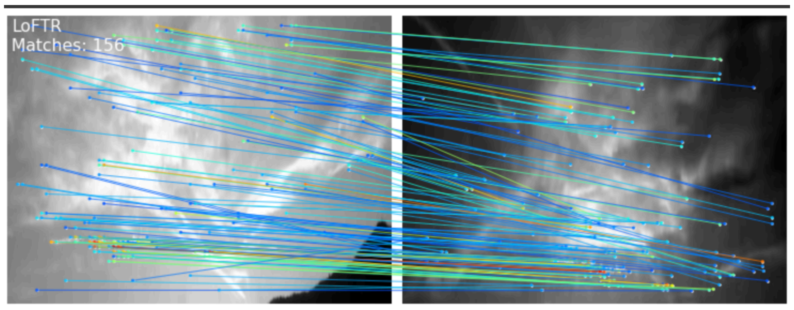


Figure 8: Pixel Matches with LoFTR

Each line drawn between two pixels is a correspondence between the two images.

When LoFTR fails for translucent figures with undefined features, manual annotation of pixel correspondences can be used, as shown in Figure 9.

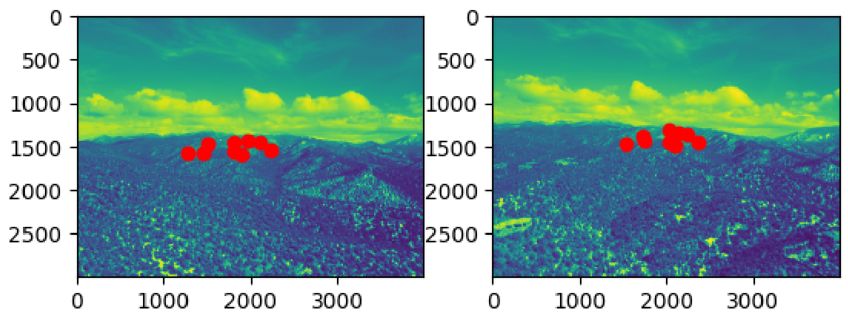


Figure 9: Manually Annotated Pixel Matches

Pixel matches were manually selected instead of utilizing LoFTR and SAM.

Most of the time, the visual media provided by smartphone cameras isn't static; instead, most of the information provided spans thousands of frames in a video format. Running both SAM and LoFTR or requiring manual annotations to track the dispersion of a smoke plume for each frame would be impractical. Therefore, after pixel correspondences have been identified in a singular frame, the Lucas–Kanade method - a method for tracking the movement of pixels in a video - is run on an initial set of point correspondences to track their location in each subsequent frame (Lucas and Kanade, 1981).

Sparse Depth and Depth Completion with DeLaunay Triangulation

Having a pixel match implies that it is visible in images A and B, providing information for triangulation. Two lines extending outwards into infinity, defined as $\vec{y_1}(t) = \vec{p_{c1}} + (\overrightarrow{p_{\text{pix1}} - p_{c1}})t$ and $\vec{y_2}(s) = \vec{p_{c2}} + (\overrightarrow{p_{\text{pix2}} - p_{c2}})s$, where p_{pix} is the position of the pixel calculated by the projection matrix at an arbitrary radius, and p_c is the position of the camera, is drawn between the two cameras and the two corresponding pixel matches. Since the calibration algorithm isn't infinitely precise, there are bound to be some errors, implying the two lines will not intersect. Thus, a vector $\vec{P}$ is drawn between $\vec{y_1}$ and $\vec{y_2}$ such that

$$\vec{P} \cdot \vec{y_1} = 0 \tag{13}$$

$$\vec{P} \cdot \vec{y_2} = 0 \tag{14}$$

To derive the location of P, s_o and p_o are derived from equations (7) - (10), scaling $\vec{y_1}, \vec{y_2}$ accordingly. The location of the pixel can be anywhere on line P; however, the assumption that the two cameras are equally inaccurate leads the pixel's location to be at the middle of the vector, at a point defined at p, defined as $p = \frac{P}{2}$. Using the described methods, the matching pixels from both images in Figure 8 formed Figure 10. A triangulation algorithm (an algorithm that creates as many triangles as possible given a set of points) - DeLaunay Triangulation - generates a 2D mesh of an object with its corresponding pixel matches to fill in the depth of the rest of the pixels (Lawson, 1972). Figure 11 shows an example of DeLaunay triangulation used on the pixel correspondences in Figure 8.

Afterward, each point in the DeLaunay mesh splits the triangle it is contained by into three separate triangles, as shown in Figure 12. An arbitrary point P has its depth calculated by the ratio using the following formula, where points A, B, C are the points of a DeLaunay-generated triangle containing point S, d_n denotes the depth of point n, and $\triangle XYZ$ denotes

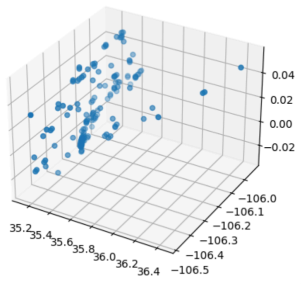


Figure 10: Sparse Depth of Cloud

Cloud triangulated with purely pixel matches.

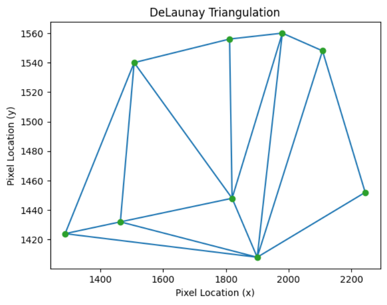


Figure 11: DeLaunay Triangulation

An example of DeLaunay being used on point correspondences.

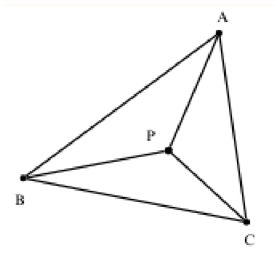


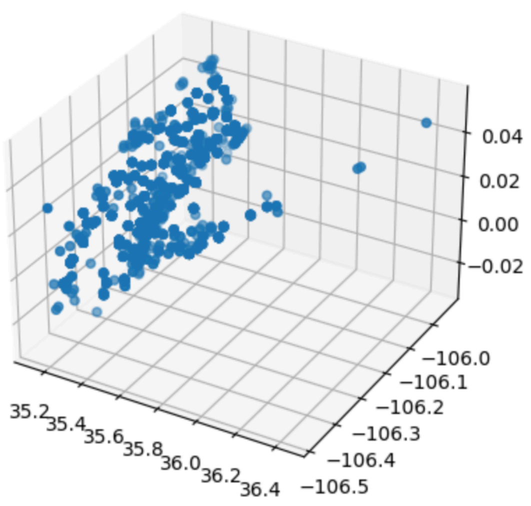
Figure 12: Triangle Split

An example of how a single match between an image and a real-world correspondence can calibrate a camera.

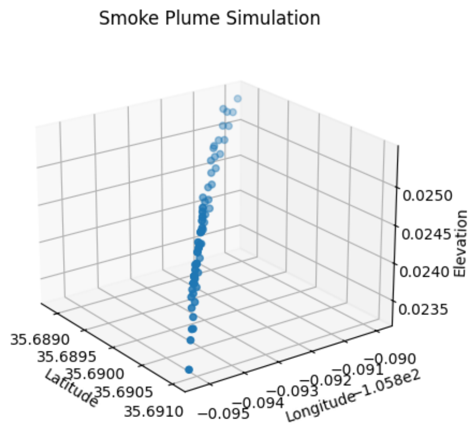
the area of a triangle containing points X, Y, Z:

$$d_P = \frac{\triangle BPC d_A + \triangle APC d_B + \triangle APB d_C}{\triangle ABC} \tag{15}$$

The results of a completed depth point cloud can be seen in Figure 13.



(a) Completed depth of cloud as seen in Figure 8



(b) Completed depth of plume as seen in Figure 9

Figure 13: Completed Depth Rendering (Granularity Reduced by 400 times for Visualization)

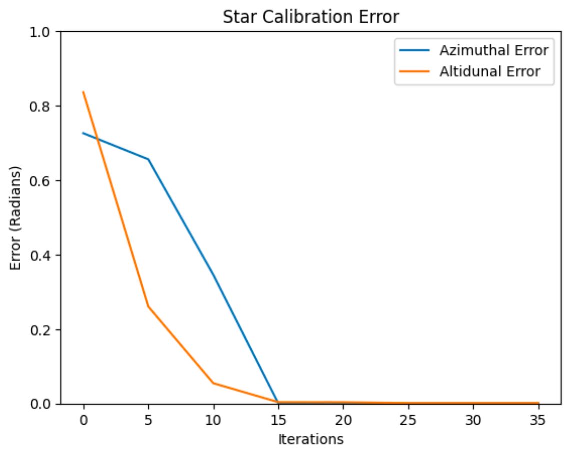
Results and Examples

Results

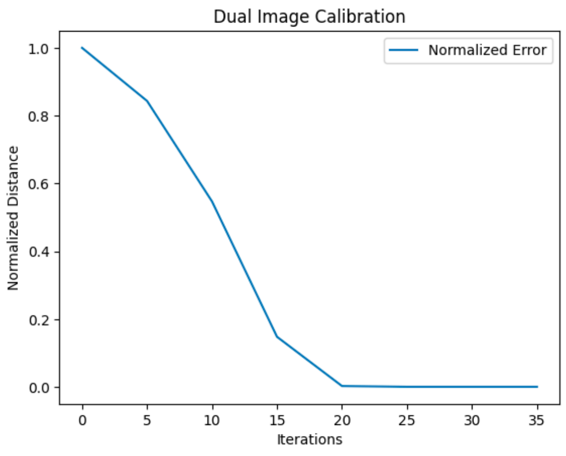
The calibration algorithm was tested with several images of stars and geographic features taken by the researcher. The collected pictures consisted of 20 images for each calibration method, all taken within the general Southwestern vicinity. For the star calibration algorithm, in under 15 iterations of the Levenberg-Marquardt algorithm, the azimuthal error between the projected position of the star and its actual position dropped from 0.725rad to $3.7 * 10^{-3}$rad, while the altitudinal error dropped from 0.835rad to $4.9 * 10^{-4}$rad. Meanwhile, in 20 iterations, the dual image calibration lowered the normalized error of 1 to 0.00346. Finally, in under 25 iterations, the single image calibration method lowered the normalized error of 1 to 0.05428.

The results demonstrate the speed at which the algorithm can solve position, rotation, and distortion parameters. However, a weakness of the above calibration method is its need for manually-annotated point correspondences. On rare occasions, users might not have access to distinct geographic features like a mountain range or a building.

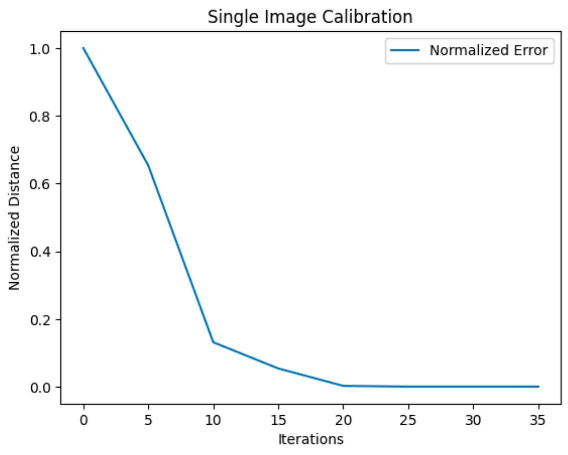
The rendering algorithm was tested with over 50 smoke plume renderings. The smoke plume imagery was taken from 10 cameras from the ALERTCalifornia camera system and through SimTable's VAPIX cameras, capturing a total of 5 different wildfire incidents (wildfires from Aztec Springs, Blue Creek, El Dorado, Del Norte, and Tulare). The photos were taken by the user using VAPIX cameras mounted on buildings in Los Alamos and Santa Fe, and a few were provided by drone footage captured by SimTable, the researcher's sponsor. The first result compares the granularity of NASA's thermal imagery using smoke plumes to the proposed algorithm's accuracy. Random clusters of triangulated points were extracted from each 3D rendering, and the average granularity - or the average distance between each particulate and its closest neighbor - was calculated.



(a) Calibration error of stars imagery



(b) Calibration algorithm of dual image matching



(c) Calibration algorithm of single image matching

Figure 14: Graph of Results

Not only was the resolution of the algorithm tested but the runtime of the algorithm was also tested. Testing the algorithm on 10 randomly selected smoke plumes revealed that it is capable of updating smoke plumes significantly faster than current technologies. Current remote sensing satellites capture wildfire progression with a 1 day time resolution, while

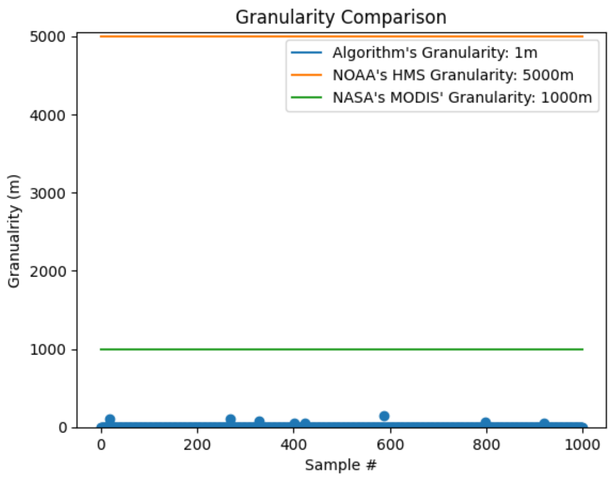


Figure 15: Smoke Plume Granularity

The average distance between each particulate from 1000 random samples.

the proposed algorithm can update a smoke plume's mesh in 5.27 seconds (Earth Science Data Systems, 2019; Data and Service, 2019). The results are shown in Figure 16. Finally,

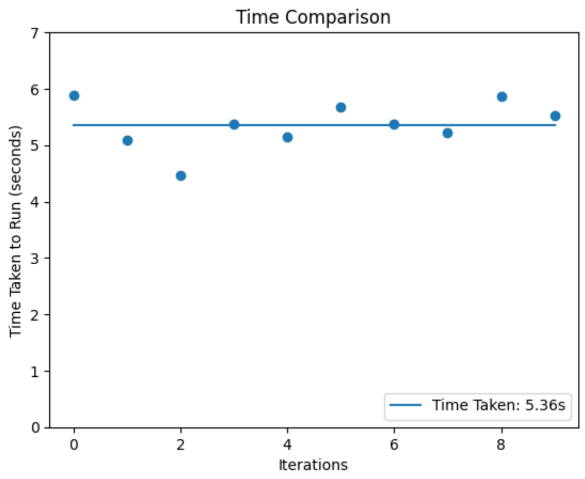


Figure 16: Algorithm Runtime

How long it takes, on average, to run the algorithm.

the accuracy of the model was calculated. Rough polygons of the smoke plume were created using aerial and remote-sensing photography and split up into individual pixels. For each smoke plume particulate, the average distance between said plume and the nearest pixel was calculated. The results can be seen in Figure 18.

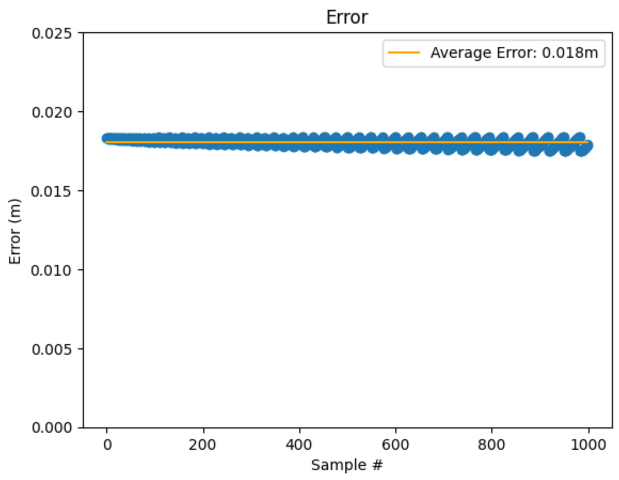


Figure 17: Algorithm Validation

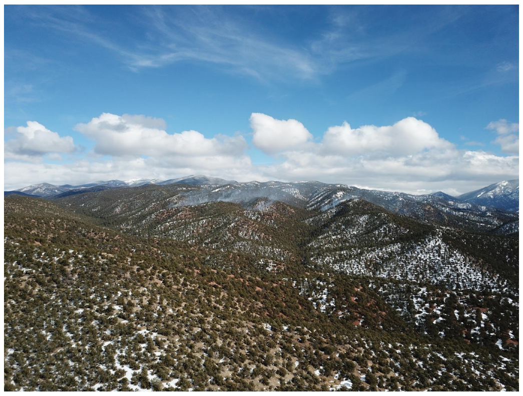
Distance of particulate from its possible values.

The method mentioned above of testing for accuracy is extremely faulty, as low-resolution images of smoke plumes cause many smoke plume particulates to be compared to the same coarse pixels.

Rendering Example

The final product should be an application on a user's device that allows them to contribute to a database of smoke plumes. Ideally, the user should take a picture of an image used for calibration and another for rendering. If another user has also taken the following steps, they should be able to render a 3D model of a smoke plume. An example of this application in a prescribed burn can be seen in Figure 18.

(a) Drone Image of Smoke Plume


(b) Second Drone Image of Smoke Plume

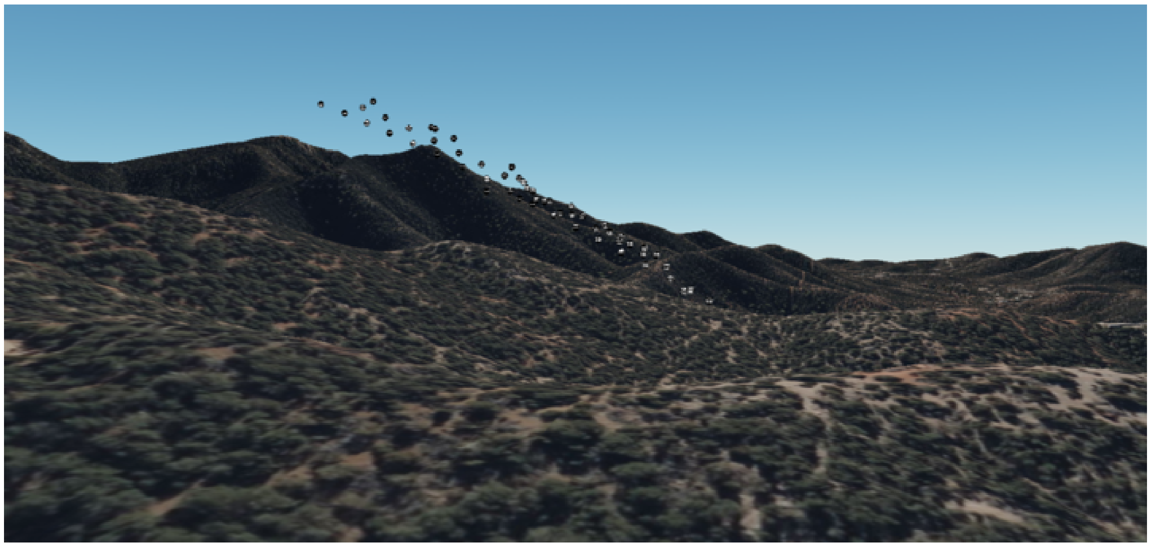

(c) Plume Render with 400x Reduced Granularity

Figure 18: Aztec Spring Prescribed Burn Rendering

Discussion/Conclusion

Discussion

As mentioned in the results, the average distance between each smoke plume particulate is, on average, 1m, implying that it can render a smoke plume at a 1x1m resolution. On the other hand, NASA's MODIS has a resolution of 1000x1000m, while NOAA's HMS has a granularity of 5000x5000m (Earth Science Data Systems, 2019; Data and Service, 2019). Most smoke plume and fire monitoring algorithms rely heavily on MODIS or NOAA's HMS, implying that the training and validation of said models rely on low-resolution surveillance data, failing to note granular geographic variations such as buildings (van Donkelaar et al., 2021; Hammer et al., 2020; Wooster et al., 2013). Thus, with a more granular database of smoke plume dispersion over a certain time interval, an improvement in the granularity of fire and smoke plume surveillance would allow for more precise prediction of fire hotspots, smoke dispersion patterns, and air qualities.

For a rapidly spreading wildfire, time resolution is crucial for evacuation and containment. In addition, rapdily changing atmospheric conditions - such as pressure, wind, or even fire perimeter spread - can drastically affect the dispersion of a smoke plume (Kelishadi and Poursafa, 2010). Therefore, a daily, low-resolution update on a smoke plume can prove insufficient for certain situations. The proposed algorithm not only overcomes the lack of granularity provided by remote sensing tools, but also updates information on the smoke plume dispersion significantly faster. In addition, since most warning systems validate their data on an hour-to-hour or even day-to-day dataset taken MODIS or HMS, a smoke plume dispersion dataset that updates within seconds can provide crucial information to communities regarding evacuation or quarantine.

Finally, the results of the algorithm's accuracy show that it can identify the location of a smoke plume pixel with an average error of 0.02m, proving that the algorithm is significantly better than current smoke plume surveillance technologies and extremely accurate. As a result, if implemented on a large scale, the model can confidently report on the location and distribution of a smoke plume, informing local communities of air pollutants that may cause them harm accurately.

Recommendations

Currently, only a theoretical prototype has been developed. In the future, all the rendering and calibration should be accessible and easy to use. To do so, the researcher needs to take two steps. First, the researcher should streamline the theoretical prototype into a single program and deploy it on a web server. Finally, the researcher should create an interface where users can take images for rendering or calibration. Note that the only costs associated with the project would be to host a webserver to run computations. Otherwise, the proposed method is completely free to implement and maintain for agencies worldwide.

The biggest limitation of this project is a potential lack of surveillance. At times, there might not be anyone with a cellphone to capture a smoke plume, especially if a fire is burning in the wild. In addition, if weather conditions limit visibility, using visual resources to detect smoke plumes may prove challenging, if not impossible. Finally, a community-sourced surveillance method may allow people to exploit the system through false visual cues or cyberattacks, which may cause casualties.

Conclusion

In conclusion, the paper presents a significant step forward in leveraging technology to address the pressing challenges posed by wildfires. While current smoke plume surveillance methods, such as remote sensing capabilities, suffer from drawbacks like speed and resolution, the proposed program overcomes such obstacles, improving wildfire surveillance technology by leveraging the abundance of cell phones. Such advantages can offer greater validation and training to cutting-edge smoke plume prediction models, causing an explosion in the accuracy of many smoke plume models.

Acknowledgement of Major Assistance

The research was aided by Stephen Guerin and his company, SimTable. He gave the researcher devices - drones and VAPIX cameras - to gather data, and advice to the student regarding testing and debugging. The research was also aided by Leah Felty (teacher sponsor), Ashli Knoell (teacher sponsor), and Cole Monical (science teacher), all of whom assisted in reviewing the paper and materials.

Bibliography

Almalkawi, I. T., Guerrero Zapata, M., Al-Karaki, J. N., & Morillo-Pozo, J. (2010). Wireless multimedia sensor networks: Current trends and future directions. *Sensors*, *10*(7), 6662–6717. https://doi.org/10.3390/s100706662

Brown, D. (1966). Decentering distortion of lenses.

Center, P. R. (2022, December). Internet, smartphone and social media use. https://www.pewresearch.org/global/2022/12/06/internet-smartphone-and-social-media-use-in-advanced-economies-2022/

Data, N. E. S., & Service, I. (2019, May). Noaa's office of satellite and product operations. https://www.ospo.noaa.gov/Products/land/hms.html#about

Dennekamp, M., & Abramson, M. J. (2011). The effects of bushfire smoke on respiratory health. *Respirology*, *16*(2), 198–209.

Earth Science Data Systems, N. (2019, August). What is remote sensing? http://www.earthdata.nasa.gov/learn/backgrounders/remote-sensing

Gavin, H. P. (2022, November). The levenberg-marquardt method for nonlinear least squares curve-fitting problems. https://people.duke.edu/~hpgavin/ExperimentalSystems/lm.pdf

Haikerwal, A., Akram, M., Del Monaco, A., Smith, K., Sim, M. R., Meyer, M., Tonkin, A. M., Abramson, M. J., & Dennekamp, M. (2015). Impact of fine particulate matter (pm 2.5) exposure during wildfires on cardiovascular health outcomes. *Journal of the American Heart Association*, *4*(7), e001653.

Hammer, M. S., van Donkelaar, A., Li, C., Lyapustin, A., Sayer, A. M., Hsu, N. C., Levy, R. C., Garay, M. J., Kalashnikova, O. V., Kahn, R. A., & et al. (2020). Global estimates and long-term trends of fine particulate matter concentrations (1998–2018). *Environmental Science; Technology*, *54*(13), 7879–7890. https://doi.org/10.1021/acs.est.0c01764

Kelishadi, R., & Poursafa, P. (2010). Air pollution and non-respiratory health hazards for children. *Archives of Medical Science*, *4*, 483–495. https://doi.org/10.5114/aoms.2010.14458

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment anything.

Lawson, C. L. (1972). Transforming triangulations. *Discrete Mathematics*, *3*(4), 365–372. https://doi.org/https://doi.org/10.1016/0012-365X(72)90093-3

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. *Imaging Understanding Workshop*.

McKENZIE, D., GEDALOF, Z., PETERSON, D. L., & MOTE, P. (2004). Climatic change, wildfire, and conservation. *Conservation Biology*, *18*(4), 890–902. https://doi.org/10.1111/j.1523-1739.2004.00492.x

Moré, J. J. (1978). The levenberg-marquardt algorithm: Implementation and theory. *Lecture Notes in Mathematics*, 105–116. https://doi.org/10.1007/bfb0067700

Muhammad, K., Hamza, R., Ahmad, J., Lloret, J., Wang, H., & Baik, S. W. (2018). Secure surveillance framework for iot systems using probabilistic image encryption. *IEEE Transactions on Industrial Informatics*, *14*(8), 3679–3689. https://doi.org/10.1109/tii.2018.2791944

NASA. (2000). https://modis.gsfc.nasa.gov/about/

OpenCV. (2021). Camera calibration. docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

Smith, K. E. (2022). https://dept.math.lsa.umich.edu/~kesmith/CoordinateChange.pdf

Sun, J., Shen, Z., Wang, Y., Bao, H., & Zhou, X. (2021). Loftr: Detector-free local feature matching with transformers.

van Donkelaar, A., Hammer, M. S., Bindle, L., Brauer, M., Brook, J. R., Garay, M. J., Hsu, N. C., Kalashnikova, O. V., Kahn, R. A., Lee, C., & et al. (2021). Monthly global estimates of fine particulate matter and their uncertainty. *Environmental Science; Technology*, *55*(22), 15287–15300. https://doi.org/10.1021/acs.est.1c05309

Wooster, M. J., Roberts, G., Smith, A. M., Johnston, J., Freeborn, P., Amici, S., & Hudak, A. T. (2013). Thermal remote sensing of active vegetation fires and biomass burning events. *Thermal Infrared Remote Sensing*, 347–390. https://doi.org/10.1007/978-94-007-6639-6_18

Wu, X., Sverdrup, E., Mastrandrea, M. D., Wara, M. W., & Wager, S. (2023). Low-intensity fires mitigate the risk of high-intensity wildfires in california's forests. *Science Advances*, *9*(45), eadi4123. https://doi.org/10.1126/sciadv.adi4123

Xie, Y., Qu, J., Xiong, X., Hao, X., Che, N., & Sommers, W. (2007). Smoke plume detection in the eastern united states using modis. *International Journal of Remote Sensing*, *28*(10), 2367–2374.

Zhu, J. (1994). Conversion of earth-centered earth-fixed coordinates to geodetic coordinates. *IEEE Transactions on Aerospace and Electronic Systems*, *30*(3), 957–961. https://doi.org/10.1109/7.303772

Detecting Exoplanets Through Transits

New Mexico

Supercomputing Challenge

Final Report

April 10, 2024

Welch Homeschool

Team Members:

Kalliope Luna Welch

Teacher:

Cindy Welch

Project Mentor:

Paul Welch

Executive Summary

The shape of a planet crossing over its star will lessen the amount of light waves that reach Earth from the star, thus making the star “dimmer.” This crossing of the planet in front of its star, similar to an eclipse, is called a transit. Professional astrophysicists have used this phenomenon to help them discover many of today’s known exoplanets, or planets from another planetary system.[1, 2] The aim of this project was to train and test a neural network (a machine learning tool) to find exoplanets through data analysis of transits. As a conclusion to this project, I discovered the different parameters that affect the accuracy of a neural network and even detected a few (prediscovered) exoplanets myself.

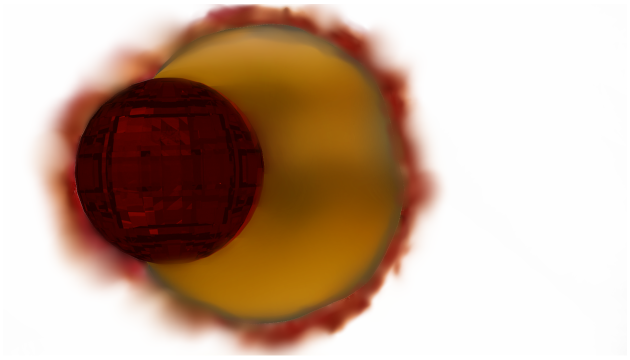


Figure 1: Model of a planetary transit.

Introduction

In 1984, the first proof of an exoplantary object was found when the Las Campanas Observatory took a picture of Beta Pictoris, a star that had a planetary disk of ice and dust surrounding it in orbit. Not long afterwards, in 1990, the Hubble Space Telescope was launched to find more such phenomena, and in 1995, Didier Queloz and Michael Mayor discovered 51 Pegasi, the first exoplanet. In 1999, HD 209458, another exoplanet, was found through the transit method that I used in this project. I used a machine learning tool called a neural network[3] to search various star data sets for any signs of a star suddenly dimming in brightness for a small period of time and then returning to its normal amplitude of light; the result of this occurrence in a set of data is called a lightcurve. Figure 2 shows a model lightcurve, the result of a transiting exoplanet.

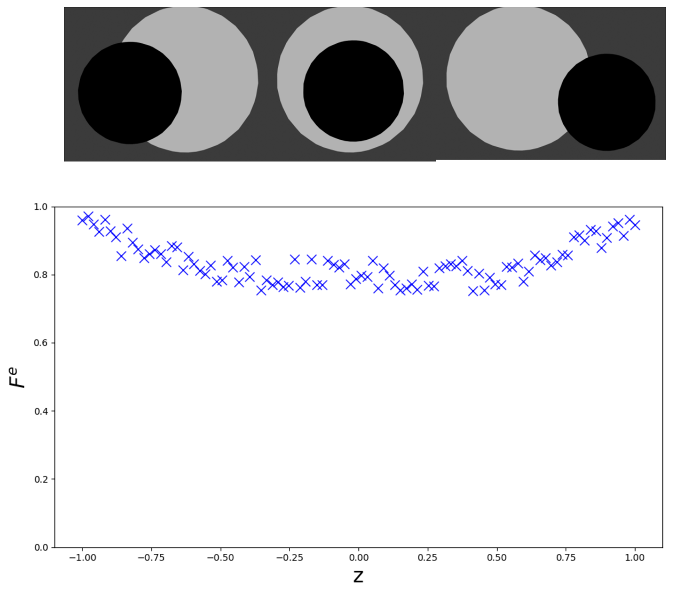


Figure 2: Diagram of a planet transiting its star (top) and the resulting model lightcurve (bottom).

A neural network was considered the best tool for this project because it will analyze the data for you, rather than forcing you to do so manually; it is therefore quicker and enables you to analyze more data than would a manual method.

Problem Statement

In this project, my goal was to discover if a neural network can detect transiting exoplanets, given certain factors. In particular, I investigated the following questions:

1) How does noise (objects passing in between the star and the telescope, as well as the telescope's abilities to gather data) affect network performance?

2) How does the size of the planet compared to the size of the star affect network performance?

3) How does the structure of the network affect its own performance?

Process

Model

My approach to this project was to first create model data, then use different datasets to train and test my neural network, and finally test my network with real data, collected from real exoplanets. To create my model data, I used the Mandel and Agol equations (Equations 1-7). In this set of equations, F_e is the fraction of light that reaches the telescope as a function of p and z; p is the ratio of the size of the planet to the size of the star, and z is the distance between the centers of the star and planet. λ_e is the amount of light obscured by the planet crossing in front of the star. I is the overall light intensity.

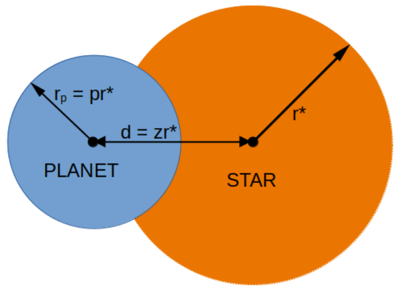


Figure 3

$$F_e(p, z) = 1 - \lambda_e(p, z) \tag{1}$$

If $1 + p < z$:

$$\lambda_e(p, z) = 0 \tag{2}$$

If $|1 - p| < z \leq 1 + p$:

$$\lambda_e(p, z) = \frac{1}{\pi}\left[p^2\kappa_0 + \kappa_1 - \frac{1}{2}\sqrt{4z^2 - (1 + z^2 - p^2)^2}\right] \tag{3}$$

where

$$\kappa_0 = cos^{-1}[(p^2 + z^2 - 1)/2pz] \tag{4}$$

and

$$\kappa_1 = cos^{-1}[(1 - p^2 + z^2)/2z] \tag{5}$$

If $z \leq 1 - p$:

$$\lambda_e(p, z) = p^2 \tag{6}$$

If $z \leq p - 1$:

$$\lambda_e(p, z) = 1 \tag{7}$$

The lightcurve in Figure 2 was generated by adding artificial computer noise to the model above (Equation 8). The noise renders the lightcurve in a more realistic fashion. Real lightcurves are convoluted, as there are many sources of interference in between the analyzed star and the detecting telescope.

$$I = F_e(p, z) + RandomNoise \tag{8}$$

Neural Network

Neural networks are made of several sets of hidden layers, as is shown in Figure 4. Each hidden layer contains a number of nodes, which make up the height of the hidden layers, a major part of my project.

To create the training set, I did the following:

a) set the input parameters, including the number of datapoints per lightcurve and the number of lightcurves;

b) selected random values for p and noise;

c) used the model to generate lightcurve data.

If the data was a true transit, I would set a flag to mark the data as 'true transit'; on the other hand, if it was not a true transit, I would set a flag to mark the data as 'noise'. In both circumstances, the next step would be to write the data to a file.

To train my Pytorch neural network, I followed the procedure below:

a) defined my neural network, i.e., the number and the height of the hidden layers;

b) inputted my training dataset;

c) fed my data into the network's input layer;

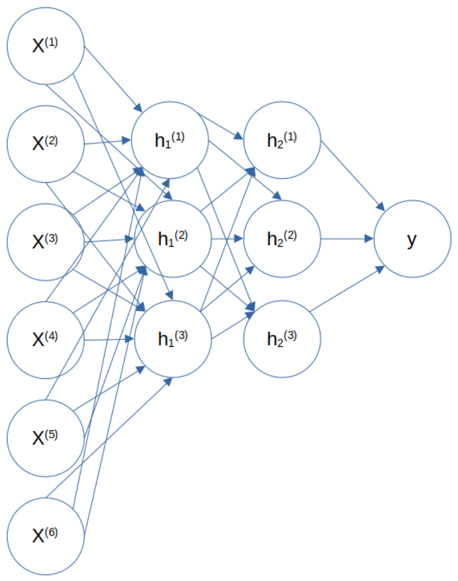


Figure 4

d) passed the data through each hidden layer;

e) obtained an estimate (i.e., does the data contain a lightcurve or not?).

If the network calculated an error across the set, it would backpropagate and optimize the parameters until the answer was correct.

Finally, I tested my neural network with real data collected from actual exoplanets, which I gathered from the AAVSO, or the American Association of Variable Star Observers.[14] Figure 5 is an example of a lightcurve covered in interfering noise. Can you tell the difference?

Results and Significant Achievements

First, I tested my neural network and made three plots of my network's accuracy based on certain parameters. Figure 6 shows that increasing the height of the network's hidden layers also increases its accuracy, up to a certain point where the accuracy stabilizes. Figure 7 shows that a higher noise factor creates a lower accuracy. One axis of Figure 8 is the accuracy of the network, another is the planet to star size ratio (p), and the third is the noise factor. In this example, my network's hidden layers had a height of 20; it was trained with 100,000 training points. Although the training sets mostly focused around $p \leq 0.2$, it was able to detect planets with larger p values, as well. The amount of noise heavily

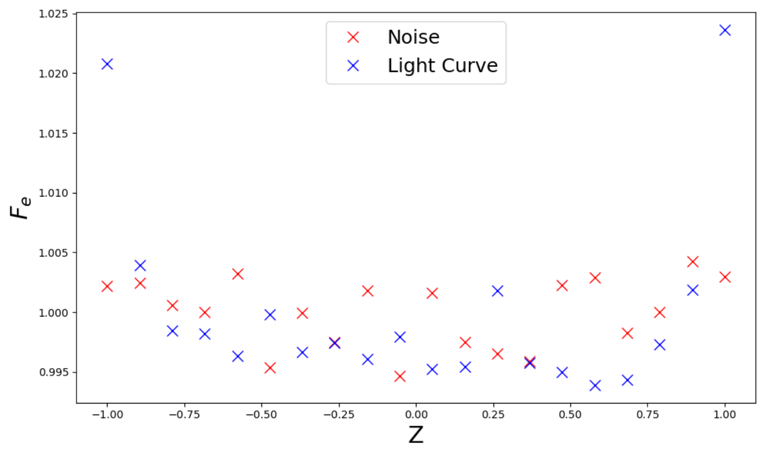


Figure 5

impacted the network's overall accuracy; as in Figure 7, increasing the noise value decreased the network's accuracy by quite a bit.

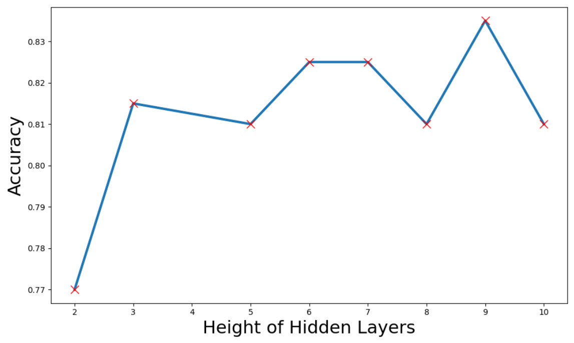


Figure 6

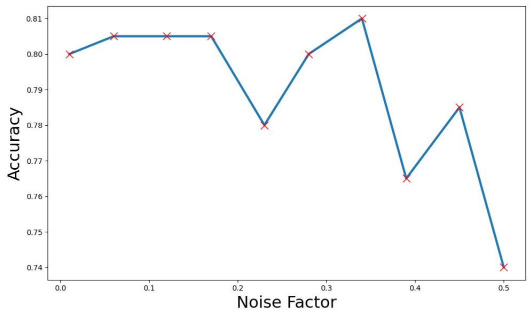


Figure 7

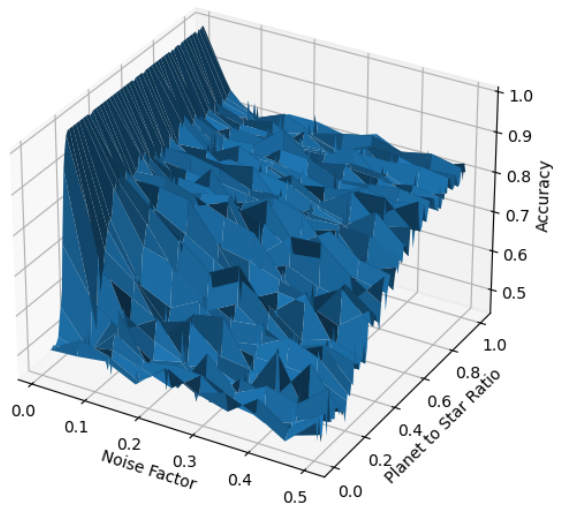


Figure 8

Figure	Height of hidden layers	Noise Factor	p
6	2-10	0.01-0.5	0.1-1
7	3	0.01-0.5	0.1-1
8	20	0.01-0.5	0.1-1

Table 1: Neural network test data parameters.

After testing my neural network, I used it to detect four real (but prediscovered) exoplanets named TrES-2b, HAT-p-68-b, Qatar 9-b, and TOI 3757. To detect each of these planets, I had to go through a process called smoothing, in which I took the running average of a certain number of datapoints from the original lightcurves. After smoothing, I had to rescale the time axis of each lightcurve to fit the scale used in training my network. I then fit the data to a spline and sampled it, taking an estimate from the smoothed and rescaled lightcurve to give to the network. Figures 9 - 12 show four sets of lightcurves for each of the four exoplanets. Each set shows the original lightcurve, followed by the result of the smoothing and rescaling process. These exoplanets were only detected by my network after the smoothing process, showing what an important part of the procedure this is.

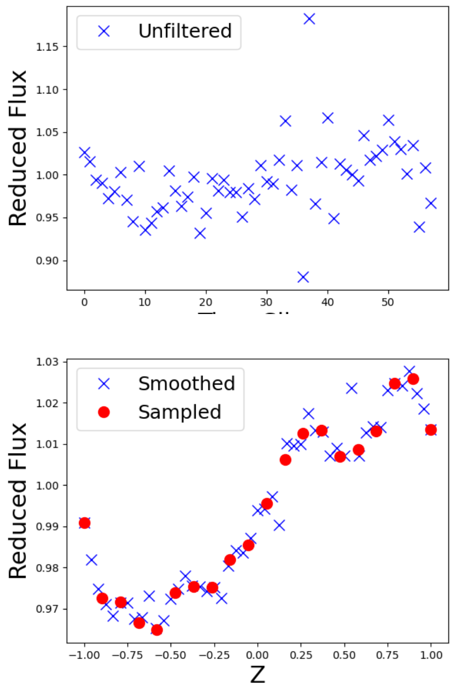


Figure 9: TrES-2b before (top) and and after smoothing (bottom).

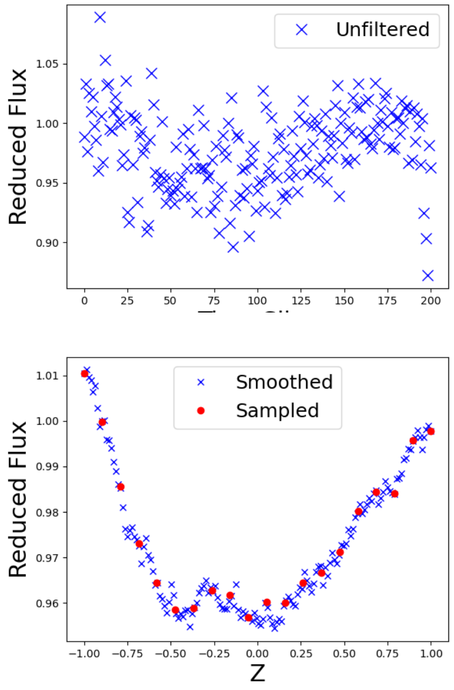


Figure 10: HAT-p-68-b before (top) and after smoothing (bottom).

As shown in Table 2, the detected exoplanets ranged from being 1.36 times bigger than Jupiter to being 1.009 bigger than Jupiter. They were discovered in time periods that ranged

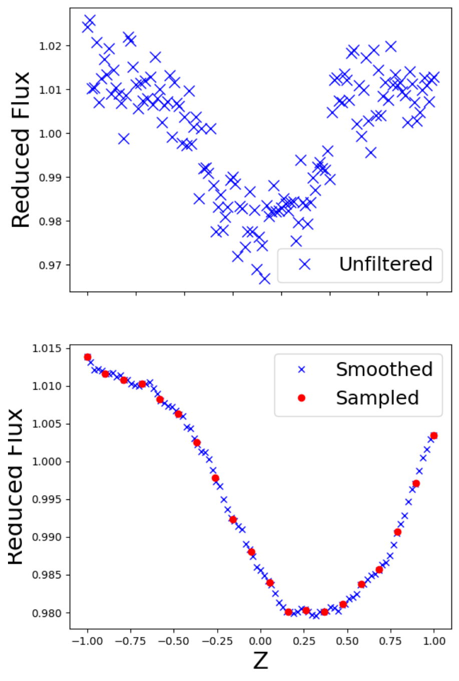


Figure 11: Qatar 9 b before (top) and after smoothing (bottom).

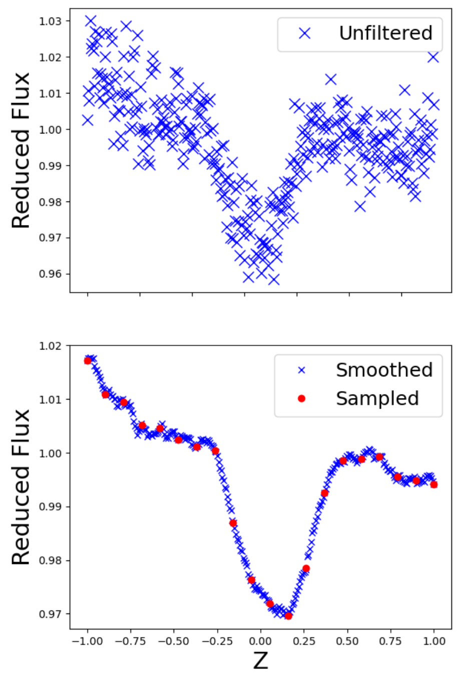


Figure 12: TOI 3757 b before (top) and after smoothing (bottom).

from 2006 to only recently in 2022. They ranged from being 591 to 702 light years from Earth. These are real exoplanets that my neural network has successfully detected. Of course, there were also some exoplanets that my neural network did NOT detect, even after

Exoplanet	Size compared to Jupiter	Distance from Earth (light years)	Discovery	Detecetd by NN?
TrES-2b	1.36 x	702	2006, transit	only with smoothing
HAT-p-68-b	1.072 x	659	2020, transit	only with smoothing
HD 189733 b	1.13 x	64	2005, radial velocity	not detected
Qatar 9 b	1.009 x	697	2019, transit	only with smoothing
TrES-3b	1.336 x	754	2007, transit	not detected
TOI-270 c	0.21 x	73	2019, transit	not detected
TOI-3757 b	1.071 x	591	2022, transit	only with smoothing
WASP-12 b	1.937 x	1,393	2008, transit	not detected

Table 2: Exoplanet information.

the smoothing process. These exoplanets are named Wasp-12 b, TrEs-3b, TOI 270, and HD 189733 b; their lightcurves are shown in Figures 13-16.

Figure 13 shows the lightcurve of Wasp-12 b. This is a significant planet, 1.973 times bigger than Jupiter, 1,393 light years away from Earth, discovered in 2008, and literally being torn apart by it's star's gravity. I was, therefore, very disappointed by the fact that my neural network did not detect this exoplanet. This is probably due to the fact that the data shows only a fraction of the lightcurve, whereas my network was trained to detect full curves. The same is true for Figure 14 (HD 189733 b, 1.13 times the size of Jupiter, 64 light years from Earth, discovered 2005) and Figure 15 (TrES 3-b, 1.336 times bigger than Jupiter, 754 light years, discovered 2007.) As for Figure 16, TOI 270-c, I infer that a cloud or other object crossed in front of the telescope during the observation of this exoplanet, and thus made the curve strangely marred.

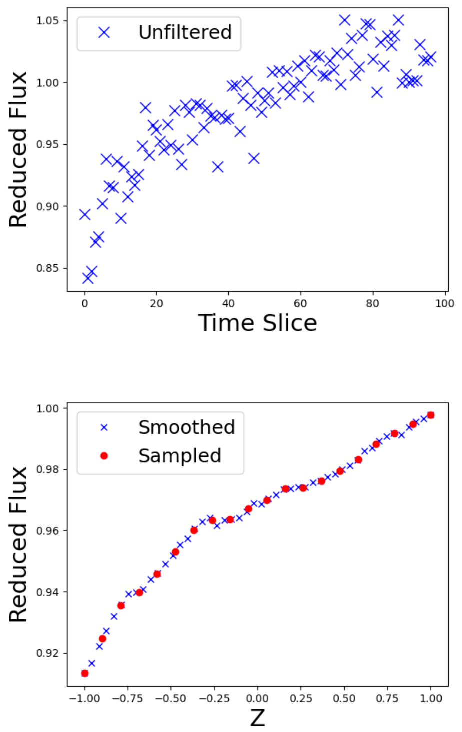


Figure 13: Wasp-12 b before (top) and after smoothing (bottom).

In this project, I have concluded how noise, p, and the structure of the network itself affect its performance. I have detected real (albeit prediscovered) lightcurves from planets outside of our solar system and tested the limits of what I could not detect. But most importantly, I have now tested and trained a tool that can discover real exoplanets.

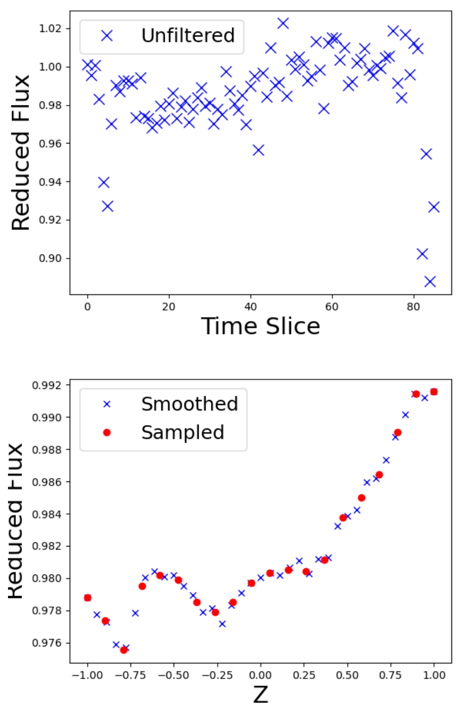


Figure 14: HD189733 before (top) and after smoothing (bottom).

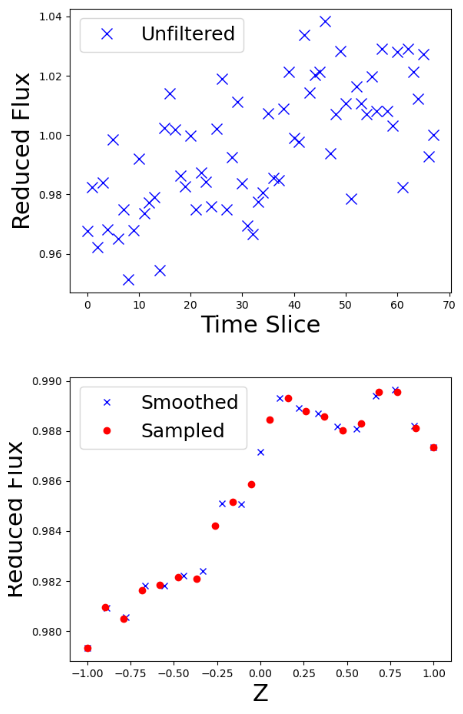


Figure 15: TrES-3 b before (top) and after smoothing (bottom).

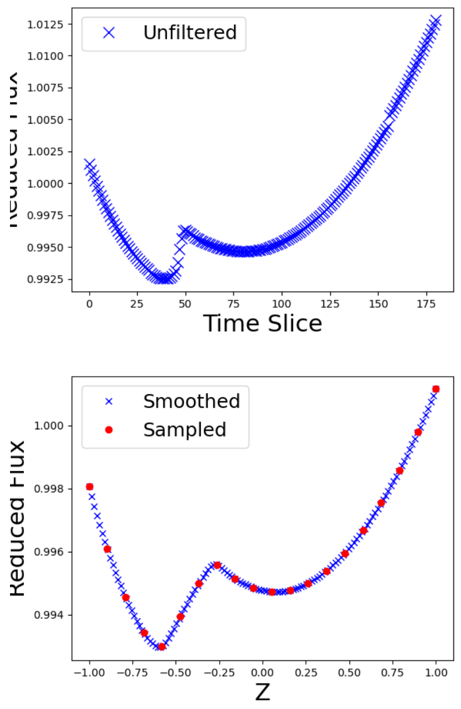


Figure 16: TOI 270-c before (top) and after smoothing (bottom).

Conclusions

From this project I was able to conclude that a neural network's ability to detect exoplanets depends on the planet system and the neural network's structure. Increasing noise decreases accuracy, whereas increasing the height of the hidden layers increases accuracy, up to a point. Larger planets will give higher network accuracy, as does smoothing the data. In the future, I hope to test my network with additional real data and find an undiscovered exoplanet using my neural network.

Appendix

My code for generating model lightcurve data is given below. All plots were made with MatPlotLib[15].

```
import numpy as np


def K0(p,z):
    term1 = p*p + z*z - 1
    term2 = 2*p*z
    return np.arccos(term1 / term2)


def K1(p,z):
    term1 = 1 - p*p + z*z
    term2 = 2*z
    return np.arccos(term1 / term2)


def lambda_e(p,z):
    if 1+p < z:
        return 0
    if (np.abs(1-p) < z) and (z <= 1 + p):
        term1 = p*p*K0(p,z)
        term2 = K1(p,z)
        term3 = 0.5*np.sqrt(4*z*z - (1 + z*z - p*p)**2)
        return 1/np.pi * (term1 + term2 - term3)


    if z <=1-p:
        return p*p
    if z <=p-1:
        return 1
    return -1000


def Fe(p,z):
    return 1 - lambda_e(p,z)


#Example
p = 0.5
Nb = 2001
I = np.zeros(Nb,dtype=float)
zboxes = np.zeros(Nb,dtype=float)
dz = 2.0/(Nb-1)
noise_f = 0.1
for i in range (0,Nb):
    z = i*dz -1
    zboxes[i] = z
    z = np.abs(z)
    I[i] = Fe(p,z) + np.random.random()*noise_f
```

Acknowledgements

I would like to thank judges and sponsors of the Annual Supercomputing Challenge for their helpful advice; I would also like to thank the the judges of the local and regional NENM Science Fair for useful feedback.

References

[1] Winn, Joshua N. “The Search for Exoplanets: What Astronomers Know” *The Great Courses* (2015).

[2] Winn, Joshua N. “An Introduction to Astrophysics” *The Great Courses* (2015).

[3] Chintarungruangchai, P.; Jiang, I. "Detecting Exoplanets through Machine-learning Techniques with Convolutional Neural Networks," *Publications of the Astronomical Society of the Pacific* (2019).

[4] "Citizen Science – Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/citizen-science/, Accessed Nov. 10, 2023.

[5] "Overview / What is Exoplanet Watch? – Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/exoplanet-watch/about-exoplanet-watch/overview/ , Accessed Nov. 10, 2023.

[6] "Background Information / What is Exoplanet Watch? – Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/exoplanet-watch/about-exoplanet-watch/background/ , Accessed Nov. 10, 2023.

[7] "Exoplanets," https://afh.sonoma.edu/exoplanets/ , Accessed Nov. 10, 2023.

[8] Mandel, K.; Agol, E. "Analytic Lightcurves for Planetary Transit Searches," arXiv: astro-ph/0210099v1, (3 Oct 2002).

[9] "Quickstart-Pytorch Tutorial," `https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html` , Accessed Oct. 13, 2023.

[10] "Request an Exoplanet Observation / How to Participate - Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/exoplanet-watch/how-to contribute/data-checkout/ , Accessed Nov. 10, 2023.

[11] "How to Analyze Your Data / How to Participate - Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/exoplanet-watch/how-to-contribute/how-to-analyze-your-data/ , Accessed Nov. 10, 2023.

[12] "Resources / Exoplanet Exploration: Planets Beyond our Solar System," https://exoplanets.nasa.gov/exoplanet-watch/resources/everything-exo/ , Accessed Nov. 10, 2023.

[13] “Get Involved / How to Participate - Exoplanet Exploration: Planets Beyond our Solar System,” https://exoplanets.nasa.gov/exoplanet-watch/how-to-contribute/checklist/, Accessed Nov. 10, 2023.

[14] Lightcurve data for known transiting exoplanets were obtained from the AAVSO International Database: “AAVSO / Exoplanet Database Search,” https://app.aavso.org/exosite/, Accessed Mar. 10, 2024

[15] Hunter, J. D. “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering 9*, *3*, `https://doi.org/10.1109/MCSE.2007.55` (2007), 90-95.