Personal Projects

Herman Nikolai Scheele

This document is intended to provide insight into the various side-projects I have worked on during my studies. All projects come from my own ideas, usually inspired by the knowledge I have acquired so far through my education. I am currently in my second year of a Bachelor's in Mathematics and Computer Science at the University of Oslo. This is a study program that provides valuable analytical skills through extensive work with mathematics, both conceptually and with pen and paper. Additionally, this program emphasizes the use of digital tools, such as programming, to solve many of today's relevant challenges, especially in Optimization, AI and machine learning, where the comprehension of the underlying mathematics behind the algorithms is crucial for effective implementation. I enjoy learning new things in the classroom, but what I love the most is creating something from my own ideas. The excitement I get from realizing an idea is what motivates me to keep building, as well as to continue absorbing new knowledge at school.

Content

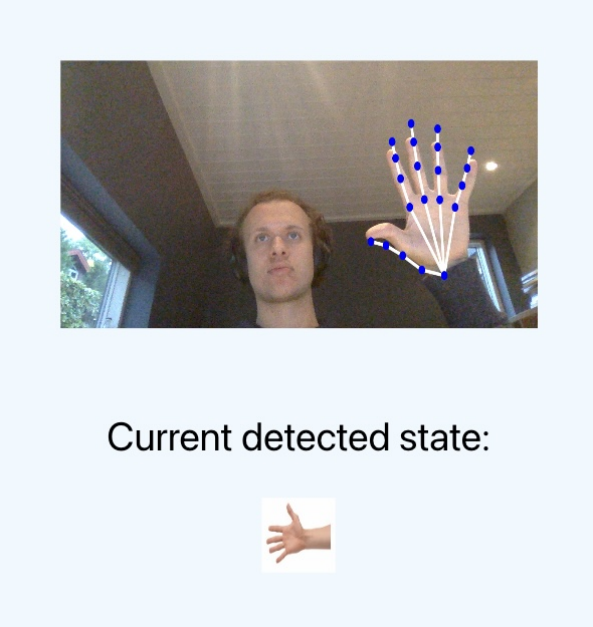

Figure 1: Paper

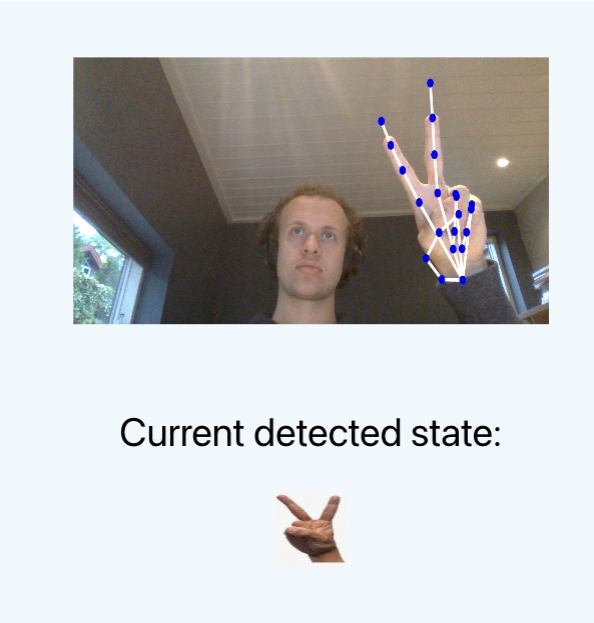

Figure 2: Scissor

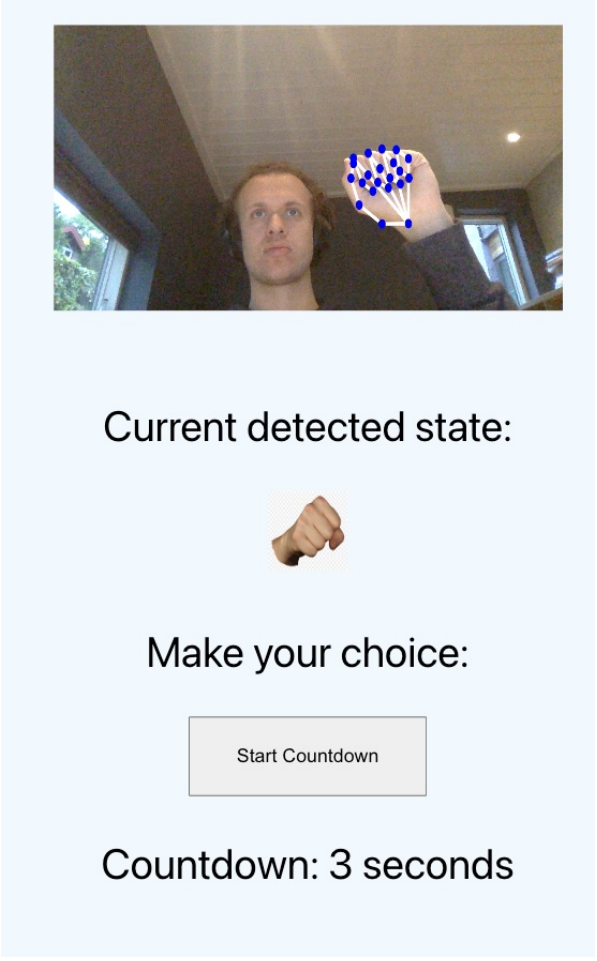

Figure 3: Rock

Description

This project is an interactive rock-paper-scissors game that uses computer vision, the TensorFlow machine learning framework, and a webcam for hand gesture recognition. The game allows players to play against the computer by showing hand gestures (rock, paper, or scissors) to the webcam. Using TensorFlow's hand pose model and finger movement recognition via the Fingerpose library, the app identifies and updates the player's hand gestures in real time. When the countdown hits zero, the player's current hand gesture is locked in as the most recent state, after which the computer randomly selects its choice, and the game determines the winner. The game is built with React and TensorFlow.js.

code: `https://github.com/hermanscheele/Rock-Paper-Scissors-AI`

Visualization of logistics data (Python)

Figure 4: With no data

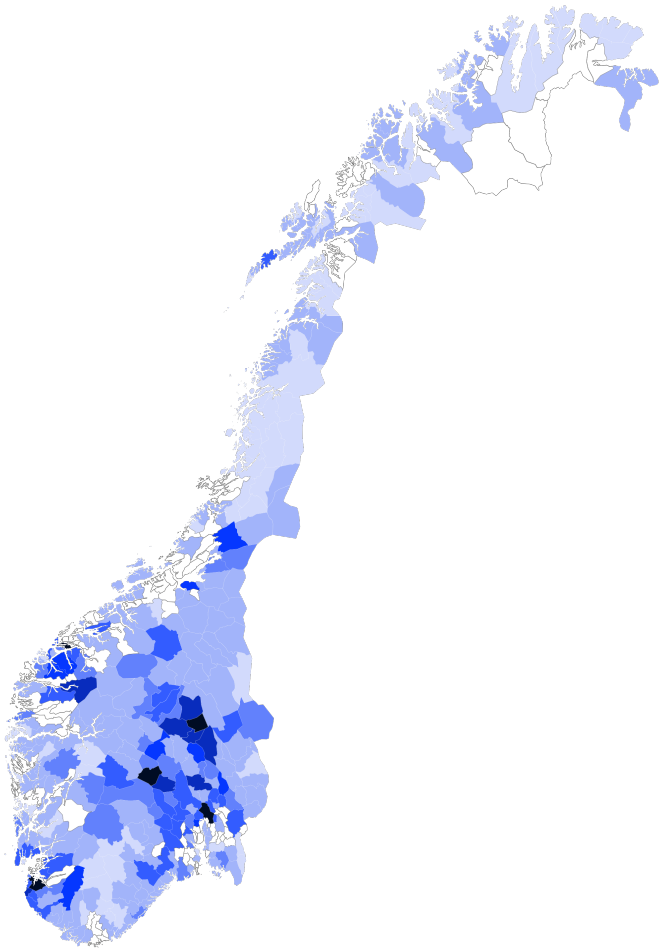

Figure 5: With data

Description

This programming project was made to solve a problem that was introduced to me by my dad. One night we discussed the issue and I came up with an idea to tackle the problem and after a couple days of intensive programming the project was finished. The program utilizes the .svg file format to plot a map based on Excel data containing order numbers and postal codes. The code determines which Norwegian municipal a given postal code is associated with and an order. Then it visualizes how orders are distributed geographically in Norway. This can be used to identify patterns in order data, such as regional differences in order volume and geographical demand. The map provides a clear presentation of logistics data that can be used for supply chain planning and analysis or marketing. The codebase in this project has already been used in a commercial setting, where it contributed to visualizing logistical patterns for a Norwegian company.

code: `https://github.com/hermanscheele/Logistic-map-visualization-Norway`

Parallel algorithm for the Convex Hull problem (Java)

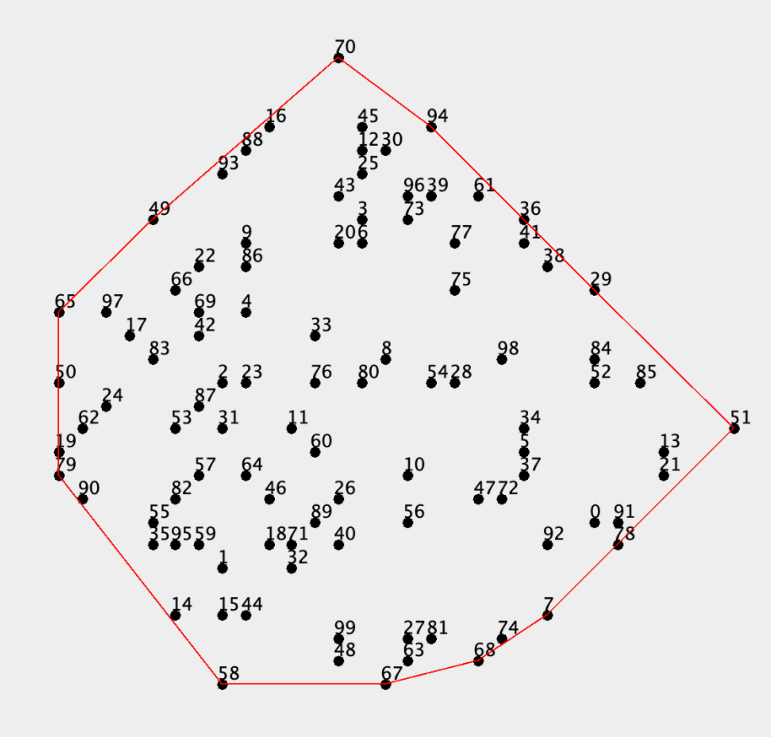

Figure 6: Convex hull

n	speedup
100	0,19
1000	0,35
10000	2,14
100000	1,54
1000000	2,32
10000000	2,31

Figure 7: Speedup with 4 threads

Description

This is a programming project where I wrote both sequential and parallel algorithms to solve the *Convex Hull* problem in Java. The parallel algorithms utilized Java's built-in thread class. The *Convex Hull* is an essential problem in computational geometry with many applications in image processing, robotics, and GIS.

Significance in Mathematics

In mathematics, the *Convex Hull* is a fundamental problem that finds the smallest convex shape that encloses a set of points. Simply put, it determines the fewest possible lines that toghether surround a figure such that all points remain within the lines. It is useful in situations where you need to create an outer boundary around a collection of objects or data.
The purpose of the project was to measure the performance of the parallel algorithm through *speedup*, which indicates how much faster a parallel algorithm is compared to a sequential algorithm when both solve the same problem. Speedup can be expressed as:

$$S = \frac{T_{\text{sequential}}}{T_{\text{parallel}}}$$

where $T_{\text{sequential}}$ is the runtime of the sequential algorithm, and T_{parallel} is the runtime of the parallel algorithm. In this case, I observed a speedup of approximately 2 for large datasets $n > 1,000,000$.

Code: `https://github.com/hermanscheele/Parallel-Convex-Hull`