Sebastian Wilkinson

[Company name]  [Company address]`s

Visualisation of a whale’s movement

Table of Contents

[Analysis 2](#_Toc63704027)

[The Problem 2](#_Toc63704028)

[Stakeholders 2](#_Toc63704029)

[Computational Methods 2](#_Toc63704030)

[Research 3](#_Toc63704031)

[Existing Solutions 3](#_Toc63704032)

[Meetings 3](#_Toc63704033)

[Solution Requirements 4](#_Toc63704034)

[Limitations 4](#_Toc63704035)

[Hardware and Software requirements 5](#_Toc63704036)

[Design 5](#_Toc63704037)

[Description 5](#_Toc63704038)

[User Experience 5](#_Toc63704039)

[Picture of user interface plan goes here 5](#_Toc63704040)

[Picture of user journey flow chart goes here 5](#_Toc63704041)

[Program Interactions 5](#_Toc63704042)

[This is a heading for a picture of all the interactions between functions and maybe structures in the program but there aren’t actually any structures except some borrowed from the standard library. 5](#_Toc63704043)

[This is a heading for all the algorithms that actually exist and maybe flowcharts for them because the examiners hopefully like flowcharts 5](#_Toc63704044)

# Analysis

## The Problem

In studying marine wildlife such as whales and dolphins, researchers sometimes use tags that can be stuck onto the animals and then either transmit data or be collected later. These tags collect large amounts of information about the movements of the animal including depth, accelerometer readings and magnetometer readings, which allow the researchers to later determine which direction the animal is pointing and moving. For more detailed pieces of data, this might be sampled many times a second over a number of hours before the tag detaches and can be collected. Other data, such as depth, might be sampled only once every few seconds over a period of a few weeks, transmitting the data via satellite every time the animal surfaces.

Due to the large volume of data collected, and the many different elements, it can be difficult to visualise the data. Researchers often use software similar to MATLAB and R to analyse the data, but it can be difficult to visualise multiple pieces of data concurrently, and these tools often suffer from speed problems when attempting to perform more complex operations and visualisations. These tools are also sometimes difficult to use to provide an interactive user interface and instead generally provide a visualisation only by running the code that cannot be manipulated without changing the code and running the analysis again.

I believe that this can be improved by providing a program that is able to display an animation of the animal that can be played similar to a video, displaying the behaviour of the animal through time. This software would allow the researchers to visualise multiple elements of the data they collect concurrently rather than one at a time, and would reduce the amount of time spent writing code to display one set of data.

## Stakeholders

This software would likely be useful to many different research groups. However, I have been in contact with a researcher at Duke University in the US and will initially tailor the software to their specific set of data.

## Computational Methods

This problem requires the analysis and visualisation of hundreds of thousands of elements of data (including heading, depth, accelerometer readings and audio data) stored on a computer’s hard drive in a short time, and the display of this data for easy examination by the end user. A computer is required to retrieve the data, and with the massive amount a computer is the only way the data could be analysed reliably and efficiently in a reasonable time.

Significant use of techniques such as parallel processing can be used to speed up initial analysis of the data and reduce the time it takes to display this on the screen.

Expand this: Problem recognition, decomposition, divide and conquer, abstraction, et cetera.

## Research

### Existing Solutions

Currently, the group I am working with uses MATLAB and R to generate visualisations such as the one seen in Figure 2 of (most recent paper about deep dives). [Reference paper]This visualisation, along with others, generally take the form of bar charts or scatter graphs.

Generating these visualisations in MATLAB requires a small amount of programming to import the data then put it on a graph, and it can then generate a static image. Changing the image requires changing the code written to display it. MATLAB also has the capability to generate graphical user interfaces and probably dynamic visualisations of data, although I have not tested this myself. However, this capability requires a large amount more code than generating basic visualisations, and so is not used often, if at all, by the group. MATLAB is also paid software, costing over £100 for a home licence and significantly more for a site licence. [Reference MATLAB website for cost and abilities]

R is a programming language designed for data handling and analysis [Reference R documentation]. It is also able to generate basic visualisations with little code, but again requires a large amount to create a GUI application. This would most likely be accomplished with a library similar to GTK [Reference GTK website], and viewing a 3D model would require further external libraries and code.

Due to the requirements in terms of code and external libraries and additional expertise in programming, visualisations of the sort that would bring together all the pieces of data into a single visualisation are out of reach for the group, meaning these existing solutions are insufficient for the problem.

In addition, both these solutions are interpreted languages. While a well-optimised interpreter can run at a decent pace, it is much slower than a compiled language [Find something to demonstrate speed difference] for operations that need to be interpreted, so most heavy data processing needs to be done with calls to compiled code. This would either be in the form of the built-in standard library or further external libraries, and code still has to be interpreted in order to make the actual decisions on what is done with the data once the processing has happened. This means that, while it may be possible to write a program using these languages to process and display the data at a smooth rate, I believe it would likely require in-depth knowledge of how the software and hardware work in order to optimise the code to a sufficient level.

Add in images of these two solutions?

### Meetings

I met with my client on the 19th of October 2020 over a zoom call. In attendance were my client, my computing teacher and myself. During the meeting, the client described her research and her basic requirements for a solution. One important point is that they are able to visualise pieces of data individually, but it is difficult to put all the data together into a single visualisation and see how it relates. I also asked about the hardware capabilities of the computers they would be likely to run the solution on. These are two or three year old desktop computers with intel core i7 processors. This means that they should support modern graphics APIs for a 3D visualisation, and they will have multiple cores and likely hyperthreading, meaning multithreading the solution will have a large performance impact [Reference spec sheets for example processor?]. My client told me that they currently use R for most of what they need, but sometimes also use MATLAB. We discussed a few different potential projects to help analyse the data, including pattern recognition or other similar applications, but settled on a 3D visualisation as the core feature because it is something that they do not have the capabilities to do at all currently.

I met again with my client on the 11th of February 2021 over google meet. In attendance were my client, my computing teacher and myself. During the meeting, we discussed the progress I had made so far and what features the client would like added to improve analysis of the data. Particularly of note was a request to graph the vertical velocity and acceleration, using estimations from the data. In addition, I asked for clarification of the sample rate so I can sync the animation with real time.

## Solution Requirements

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Requirement  No. | | Requirement | Solves | Evidence | Justification | Success Criteria |
| 1 |  | 3D animation and graphs of data | Visualise the data |  |  |  |
| 1.1 | 3D model displayed on the screen | 3D visualisation |  | Display direction and movements of the whale to the user of the system | - 3D model is rendered  - Model rotates to accurately reflect the data |
| 1.2 | Graphs of pitch, roll and yaw displayed | Individual visualisation |  | Allows the user to view the exact attitude of the whale at a point in time, and find points where the whale is doing something easily. | - Graphs are displayed on the screen  - Graphs accurately reflect the data and correspond exactly to the 3D model |
| 2 |  | Responsive and quick application | Application is smooth and easy to start |  |  |  |
| 2.1 | Fast loading times | Easy to start |  | It does not take so long to get a visualisation that the users have time to go and get a coffee. | - Under 10 seconds from loading in data to visualisation populated  - Preferably under 1 second |
| 2.2 | Animation has high framerate | Application is smooth |  | The user should be able to see the animation smoothly to easily and correctly see what the data is showing. | - At minimum frames should be rendered at the polling rate of the data  - Preferably at the refresh rate of the monitor |
| 2.3 | Any UI element that takes input responds instantly to user input | Application is smooth |  | The user should not have to wait half to over a full second for the program to respond to what they want to happen. | - Input handling takes place every frame  - Effects of input handling are visible on the next frame |
| 3 |  | Data analysis capabilities | Analyse the data |  |  |  |
| 3.1 | Data export between marks | Analyse the data |  | The user needs to be able to extract the data for certain periods to perform analysis not provided by this application | - Markers of export boundaries clear and button to set is easily accessible  - Correct range of data is exported |
|  | 3.2 |  |  |  |  |  |

### Limitations

While I have focused on visualisation of the data in this analysis, the group would also benefit greatly from automatic analysis of the data through some form of AI or machine learning to identify interesting points in the data. Some limited form of identification of dive times might be possible, but machine learning or any further analysis than this is beyond the scope of this project and would take far too long to implement.

## Hardware and Software requirements

At this stage, it is impossible to say for certain the requirements of the solution as the solution does not exist yet to test resource usage. However, I expect that, for an optimal experience, a modern (i.e. last 5 years) CPU and at east 8GB of RAM will be required as the dataset is quite large. I do not expect that there will be additional external hardware or software requirements, as all libraries used and any extra utilities needed can be packaged with the main solution.

# Design

## Description

The program will read in the depth and orientation data then display it on the screen. The orientation data will be shown both as graphs and as a 3D model oriented correctly relative to a set of axes such that the camera can be rotated around to view the orientation and movement of the whale. The depth data will be shown only as a graph. Each graph will be accompanied by a number showing where the whale currently is and an indicator on the graph for the current time. Heading may be displayed as the rate of change of heading or as the absolute heading. Depth will be accompanied by both the velocity and acceleration. These will have to be calculated as they are not part of the provided data.

## Decomposition

|  |  |  |
| --- | --- | --- |
| Input | Data files (depth, yaw, pitch, roll vs time | Files are assumed to span the same time range |
| Libraries |  | Prespecified |
| Output | Graphical output  Exported selection of data | Depth/time  Yaw/time  Pitch/time  3d whale  Velocity/time  Acceleration |
| User interactions – to modify display | Selection of data point  Zoom in on section of data  Export |  |

Graphical user interface – initial screen

Import data

Data needs to go on screen

User selects data for each type, or loads cache

Analysis is performed on data

Libraries manage drawing graphs

Display shows graphs

User choices modify display – eg 3D display, play through sequence, zoom in

User chooses – export data

## Class diagram

## Data table

## Entity relationship

## Data flow diagram

## Algorithms

## Key variables and structures

## User Experience

### Picture of user interface plan goes here

Graphical user interface, diagram

Description automatically generated

### Picture of user journey flow chart goes here

Diagram

Description automatically generated

## Features that make the system more useable

## Program Interactions

### This is a heading for a picture of all the interactions between functions and maybe structures in the program but there aren’t actually any structures except some borrowed from the standard library.

### test plans for algorithms and iterative development

## Post-development test strategies

Diagram

Description automatically generatedDiagram

Description automatically generated

# Development

Development of the project was split into three sections.

In the first section, I worked to get all the libraries to work together. In the second section I developed an initial iteration of the project with most of the base features. I then asked my client for feedback. In the final section, I acted on this feedback to polish the application and add a few new features to better meet their requirements.

## Section 1

The main libraries I had to work on are BGFX and ImGui. BGFX provides a higher level interface to graphics libraries, making it much easier to draw the 3D model to the screen. ImGui is an immediate mode graphical user interface library. This means that user calls directly cause the library to render to its buffer. The problem with this is that ImGui needs a way to call whatever is being used to render. There are a number of example implementations provided by the authors of the library, but it was very difficult to find a backend built for BGFX. I did manage to find one eventually, but it did not work with all the features of ImGui that I wanted so I had to extend the backend to use these features. Mostly, this took the form of borrowing code from the example backends until things worked.

A picture containing text, monitor, screenshot

Description automatically generated

A picture containing text, monitor, indoor, electronics

Description automatically generated   
As you can see in the above screenshot, there were some issues during the development of this capability. I eventually worked out what was going wrong and fixed all the issues.

You might also notice that the cubes rendering in the images are inside out. I’m not entirely certain why that was the case, but I fixed the model in later versions so it no longer happens.

## Section 2

The aim of this section was to produce an initial iteration of the basic features of the application. Graphical user interface

Description automatically generated Shape

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedA picture containing text, monitor, screen, screenshot

Description automatically generatedA picture containing text, monitor, screenshot

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with low confidence