artevelde hogeschool
LID VAN DE ASSOCIATIE UNIVERSITEIT GENT
denk. doe. word.

Optimizing Garbage Truck Navigation

Bachelorproef voorgedragen door Jonatan Verstrate tot het behalen van het diploma Bachelor in Grafische en Digitale Media en New Media Development

Arteveldehogeschool, Hoogpoort 15 9000 Gent

Jonatan Verstraete
Grafische Digitale Media
3 New Media Development
2022-2023


BINAV

Table Of Content

Introduction

Background

In 2021 I worked a summer job as a garbage collector in Ronse.
The main company responsible for trash collection in East-Flanders is called Renewi. My job was pretty straightforward: with a colleague I got to ride on the back of the garbage truck, jumping off to collect garbage cans and tip the contents into the truck.
One morning I arrived at work early. All the drivers and planners sat huddled round the table, discussing todays workload. There seemed to be some kind of trouble going on, and sheets of paper were spread out over the table. The papers looked to be a collection of maps in various colors and shapes, some stained and creased and old, others newer looking.
Later, a colleague explained these are annotated garbage routes, hand drawn by the local veteran drivers. These people know every route and its particular conditions inside out, and their knowledge is invaluable to the crew, and by extension their employer.
Intrigued, I asked our route planner for such a map. She refused to let me borrow, copy, or even photograph the ones in her possession. But she told me something fascinating - these maps are crucial, and serve as the cornerstone of their company.

Being a student of interactive media development, I obviously saw opportunity here: to replace the old hand drawn and printed maps by something more efficient, like a routing app.

Cellphone/Smartphone usage while driving is not a good idea, but the garbage truck drivers have a unique feature that sets them apart: their frequent stops are the ideal backdrop for the driver to interact with a simple app. Later in this paper you will find that during the development of the product, there was a solution which was found for the end-users use of the app while actively driving.

Objective

The goal is to optimize the driver experience by digitalizing their hand drawn maps and routes. This can be achieved by offering an app with features that allow the user to easily navigate on an interactive map of their active route.
The users of this app will be the garbage truck drivers

Thesis structure

This thesis is divided in 2 main parts, and a conclusion.
The first part details the research in a practical sense, as seen from the end user-experience, under the title ‘Empirical Research’
This because of the need to discover which features to be essential to create this ‘niche’-navigation app
After theorizing on this, practical research had to be done to determine which of the driver’s daily challenges could potentially be solved by in-app features.

The Technological research section explores the relevant technology out there, to see what resources could potentially be used for the creation of this product.

The second chapter is by all means the largest - The production process.
Using the data from my research, the essential features are distilled for the app.
There is a separate section for end-user experience, as this plays an important role.

Then the scope gets defined, as well as the target audience,
This is followed by technical implementation, and here-after: conclusion.

1. Preproduction

1.1. Empirical Research

Introduction

As I was fortunate to work among my target audience, I was able to gather a lot of valuable information about problem statements while working. I also interviewed some coworkers, outside of working hours, which are included in the sources.

So the information in this chapter is based on interviews with colleagues, and my own observations while working amongst them. In the following paragraphs, I will illustrate each category with actual scenarios, highlighting key features that I believe should be included in the app.

Scenario 1: Infrastructure

The bulky garbage trucks need to navigate creatively; for example entering certain roads backwards because of their narrowness. Years of experience adds little knacks: you do the streets of subdivision R in this particular order, then you double back less often; or knowing that on rainy days the only driveway where you can turn on point X of route C gets dangerous so you need to detour via Y point.
Additionally, Belgium holds international fame for its ‘abominable’ road infrastructure (the Guardian, 2020), creating impossible traffic conditions .
Algorithms such as Google maps don’t provide a lot of support under these unique circumstances.

Description

During our morning route, our garbage truck encountered an obstacle on a narrow street caused by roadworks that were not anticipated. As a result, we could not move forward, and our only option was to reverse. By the time our driver accepted our situation caused a traffic jam with two other garbage trucks and many cars behind us. Coordinating the reverse maneuver of our three trucks required a lot of time and effort. Ultimately, the waiting cars had to pull into the adjoining fields to allow the trucks to pass.

Possible solutions: alert feature

If our colleagues had known about our standstill, they could have reversed in time to avoid joining the traffic jam. Methods such as texting or calling may not be preferred by the drivers, as shown in the scenario above. A map-based feature similar to Google’s Waze, to alert other drivers to obstacles or unexpected standstills, would be a valuable addition to the app.

Scenario 2: Care Packages

At the end of the day all streets assigned for the day must be cleaned. The crew operates as a team and often assists each other, even if it means taking on extra work. This unwritten rule is highly respected.
If a team finishes their work early, they often use their extra time to help out their colleagues.

Description

Garbage-collecting companies usually work with different teams on different locations.
In this scenario, the A-team is on their way back to the depot. On their way, they see PMD bags have not been picked up yet on Violet street. The collectors jump off the truck and swiftly collect the PMD bags into one spot . This 'care package' is left on the side of the road for their colleagues to pick up. The A-team does not communicate the existence of these 'care packages', expecting they will be noticed by the crew on their routes.
Hassan, the B-team driver, is in a bit of a rush because they are behind schedule that day. He notices the next street on his route, Violet street, is clear of garbage. Sweet, he thinks, somebody else decided to do this bit today. I'll take a left here. Taking this turn, Hassan misses the 'care package' waiting for his team at the end of Violet street. Needless to say this creates issues, especially in summer when garbage attracts flies and vermin.

Possible solutions: pins

A simple pin on a map. An advanced scenario makes the pin disappear when the team 'owning' that route has been detected nearby the pin.

Scenario 3: Language & Communication

The city of Ronse is situated on the border between Walloon-region and Flanders, where they speak French and Dutch respectively. Employees often speak only one of these languages.
The Renewi workforce includes many first, or second generation immigrants who are still in the process of attaining native speaker status in any of Belgium's official languages.

Description

It was the end of our working day and we had just finished our route. A colleague, Stevie, phoned our driver, Bruno, asking for help. Such requests happen quite often. Stevie had partly done his route, but for some reason was unable to do one particular area, and wanted our help. He rattled off a list of streets. Bruno took a pen and tried to mark them on his map.
An argument started: the drivers were unable to settle on the correct street names. Bruno speaks heavily accented Dutch, while Stevie only communicates in French.
In the end Bruno gave up and we went back to the depot. On arrival we heard Stevie didn't finish his route, as he assumed we were doing it. We had to drive back and finish up because Stevie and all the crew had left for home.

Possible solutions: Transfer Feature

Not only do we need to bridge the language issue, but it looks like live adjustments of routes should somehow be tracked as well.
A system that allows a driver to send a request for help to a colleague, and they can respond by simply clicking 'accept' 'reject' or 'suggest alternative'.

Scenario 4: Time & Progress tracking

At Renewi Ronse, the crew is paid hourly. The paid hours only begin once they start their route. Travel from the depot to their route is not paid. Logging happens manually: the driver fills out the crew names and start time on a paper form. When the route has been completed, the driver repeats the process including the end time. The driver is responsible for submitting this form to the administration after work. Crews occasionally abuse this system by taking unauthorized breaks or ‘extending’ their working time. The paper form itself is a liability: drivers often mislay or lose their forms, forget to fill them out on time, or even forget to submit the form to the administration.

Description

Sometimes when a team is done with their shift, they are asked to help out another team, this is very common, and is well payed, so is more of an unspoken rule than a request.
But this new ride also requires a new form with the names of the workers. But the driver an the other works forgot about the second form, and as a result did not get payed for their full 3 extra hours.
This was only noticed as one of the workers was a new student, who noticed he did not receive the hoped for extra paycheck.

Possible solutions: Progress tracking

An automated progress tracking system, tracking how much of the route has been done, when it started and when it is finished. This way the team instantly start working and afterwards, drive straight back to the depot.

This would also give extra benefits, like data analytics for route optimization, and an oversight for administration so they can see in advance if someone will require extra help.

1.2. Technology Research

A. Introduction

The market is flooded with a plethora of navigation systems, however, when it comes to garbage trucks, their specific requirements need to be taken into consideration.

As such, it is imperative to focus on software that supports one or more of the key areas described earlier. It is not enough to simply choose any navigation system as it may not cater to the specific needs and requirements of garbage trucks.

Therefore, a thorough evaluation of available options is necessary to identify the software that can meet the unique needs of garbage truck drivers. This approach will ensure that the navigation system selected is efficient, effective, and tailored to the specific requirements of garbage truck operations, ultimately resulting in a more streamlined and productive workflow.

B. Existing navigation applications

Map-pak

About
Website: https://soft-pak.com/map-pak
Price: upon request
Location: Worldwide, based in California.
Target market: garbage truck companies
Route management and optimization software for refuse haulers.

Specifications
- Uses Google Maps
- Visual representation of routes in different colors
- Updates new streets automatically (no software to load)
- Uses device location service directly
- Visually optimize, move and resume stops on routes
- Create geofences and receive email alerts when drivers enter and exit the vehicle

Analysis
Map-Pak is designed to help organizations manage their fleet. Built on Google Maps, which provides a real-time view of all routes and their stops, enabling dispatchers to make real-time changes (DoverESG, 2023).
One the features that I might be able to use is the list of all routes which can be assigned to the drivers.
However, this software is not open source nor provides any documentation, and is not free software to use, so can not be directly used this for this project.

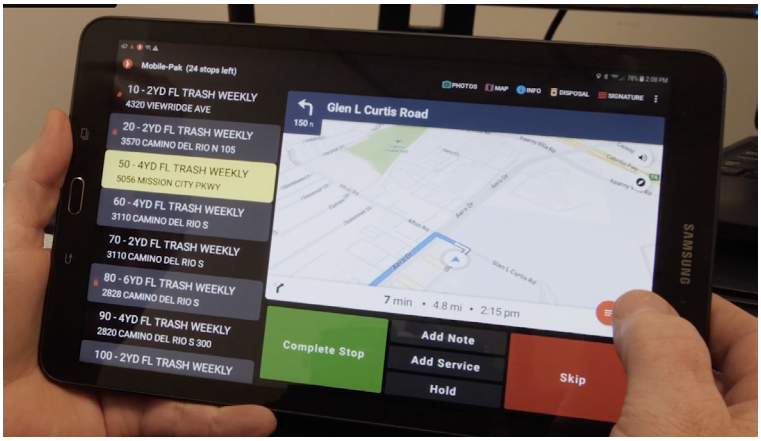
Mobile-pack

About
Website: https://soft-pak.com/mobile-pak
Price: upon request
Location: Worldwide, based in California.
Target market: garbage truck companies
Software and app for drivers.

Specifications
- Real-time GPS tracking using Google Maps
- Scheduled and on-call work is immediately sent to drivers
- Mapping of local streets and highways
- Photo capabilities for overloaded/blocked containers
- All driver activity is recorded
- Recorded service time for each stop
- Display of list of possible routes
- Option to stop routes

Analysis
Mobile-Pak is designed for garbage truck drivers.
The application offers features such as route optimization, real-time GPS tracking, and two-way communication with dispatchers.

The user interface is very technical; showcasing a lot of detail. This amount of detail is distracting for an active driver. Strong features include: dark mode, minimalist product design when it comes to color and shapes. Furthermore, the size of the buttons is conveniently large; improving ease of operation.



[Screenshot of Mobile-pack ad]. Retrieved February 2023, from https://vimeo.com/733663235

Sygic

About
Website: https://www.sygic.com
Github: https://github.com/sygic
Price: $25/month
Location: Worldwide, based in Slovakia.
Target audience: Personal, families, corporate fleets.

Specifications
- possibility for offline 3D map navigation
- synchronization of route on Android with Browser extension
- Chooses route based on vehicle (height, wight..)
- Head-up Display
- Real-times information of speed limit & speed camera warnings,
- dynamic lane assistant
- checks route and vehicle for emission zone
- uses geofencing to send notification when a vehicle leaves a certain area

Analysis
Sygic can be used by individuals and businesses alike. The app provides a wide range of features like, geofencing or the possibility to integrate third party software which let's for example synchronize your contacts with the app. They also provide the option to customize the user interface.

Although Sygic is not entirely suitable for garbage trucks, it does provide interesting features such as turn-by-turn navigation, real-time traffic updates and offline maps.

Sensoneo - Drivers Navigation App

About

Website: https://sensoneo.com/
Price: on request
Location: Worldwide, based in Slovakia.
Target audience: Smart Cities, Waste Collector, EPR & PRO organizations, Factories & Industries.

Specifications
- In-depth efficiency analysis and statistics
- A proprietary route planner engine
- Fixed routes, or routes on demand, displayed in calendar schedule
- Calculate cost, time and distance of collection
- Navigate drivers via the Android app
- Track collection vehicles on the road
- Uses GPS localization in Android mobile devices
- Ability to set vehicle capacity, bin capacity, bin type, work schedule... etc.

Analysis
An app specifically for garbage truck drivers, using the Sensoneo software ecosystem.
It offers distinctive features like: real-time monitoring of fill levels, route optimization, step-by-step-navigation.
However, this software can not be used for this project as the Drivers Navigation App relies on a network of apps made by their own company.

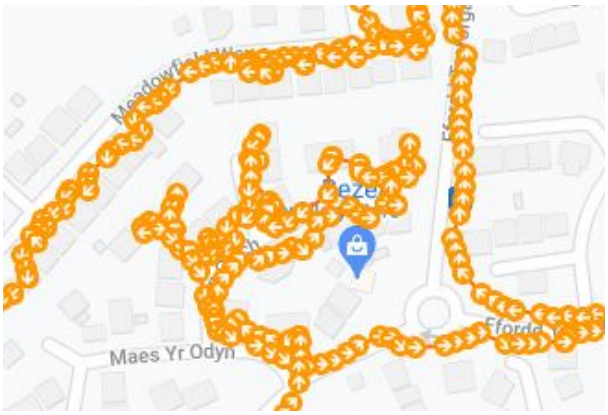
Trackit 247

About
Website: https://trackit247.com/
Price: on request
Location: United Kingdom.
Target audience: Location tracking industries

Specifications
- Offers statistical insight
- Map with overview of trucks
- GPS and GNSS (What Is the Difference Between GNSS and GPS? - Everything RF, n.d.) tracking

Analysis & Conclusion
This software looked particularly promising, as it uses an open source (developer friendly) software to display their map: Leaflet (GPS Leaflet Tracking Software for Leaflet Distribution, n.d.) and offered a demo.
So I made contact, pretending to be a company owner looking for a solution for my fleet, and they gave me a free demo account.
This gave an interesting insight into their product. Yet the product itself turned out to be neither open source, nor does Trackit 247 provide any options for developers. As such it could not use this for this project.

Trackit 247 is mainly designed for tracking trucks, but I found no option to assign routes to vehicles. Overall their methods of tracking results in rather complicated images, such as the one below.



[screenshot of Sygic demo app] from https://www.sygic.com

1.3. Requirements for design

A. Introduction

This section analyzes the requirements for the app. Through a more in-depth look at the users' needs and the (non-) functional requirements.

B. Features

Scalability Requirements
The app should be scalable to accommodate future growth in the size of the garbage truck fleet, as well as changes to the road network and other environmental factors.

Performance
To ensure efficiency, the speed and quality of this product must both be optimized to a significant degree.

Built-in features

Vehicle telematics integration:
Currently telematics are displayed on multiple screens in the cab of the truck.
This creates both confusion and distraction. The app should be able to interact with these systems. A centralized view would greatly improve workflow, and an on-demand display option could easily be integrated. Vehicle performance metrics such as fuel consumption, engine temperature, and tire pressure could then be monitored in real-time.

Data analytics capabilities
The data collected while using the app's features could, over a period of time, accumulate in a dataset that can be analyzed for route optimization, analytics, and overall efficiency of the garbage collection process.

Real-time GPS tracking
The app should provide accurate, real-time GPS tracking of each garbage truck in the fleet. This will enable administration to monitor the progress of each truck and make adjustments to the route as necessary.

Route specific requirements

Transferring roads
The driver needs an option to transfer part of their route to a college in very straightforward way. A simple click on a menu, could then replace the frustrating and ineffective communication scenario's that can occur.

Progress tracking
The possibility of tracking how much of the current route still needs to be done. This so drivers don't have to mark, or remember which part of the road they have done, so tracking progress can become a fully automated system.
This progress could optionally be shared with other drivers in order to possibly help each other out, and also to gain deeper insight with data analytics.

Offline Map
A physical map has the advantage that it does not require Wi-Fi, which can fail during harsh weather, but we can solve this by making a map offline available.

Automated routing and scheduling
The navigation system should be able to automatically generate optimal routes and schedules for each garbage truck, taking into account factors such as traffic, weather, road conditions, and the locations of pick-up points.

Optimization
With deep learning systems and the help of AI, routes can be updated with changes in e.g. road networks.

User Experience design

User Experience plays a crucial role in any successful design of an app. This chapter explores how I should customize the UX design to my target audience, and outlines features that must be included in the app to create an engaging experience for the end-user.

Imagery
Drivers should only spend short windows of time interacting with the app as their focus lies on driving.
A minimal design is needed: to limit distraction, and to enable the driver to easily switch focus between the interface and the driving. This means also selecting colors that are not too contrasting while working in the app.
Imagery needs to be easy to recognize and understand, and the icons for different actions need to be visually distinctive so they can be recognized at first glance.

Map layers
Drivers should be able to customize the map layers according to their preferences, displaying or concealing map components like trees, hills, or restaurant markers. A density selection should be a feature, allowing the drivers to choose the level of detail displayed on the map. This customization would allow the drivers to focus on the vital information they need.

Personalization
Making an app user-friendly involves personalization, which means taking into account the different preferences of users. For example, some users prefer larger fonts, less distracting colors, a dark mode, or the option change the icon of their truck.

Drive Mode
While driving, the amount of options and functions in the UI needs to be minimized. For instance, if a driver wants to customize their truck's icon, they should only do so when they are not on duty as it could be a distraction while driving.

External controls
User experience could be even more facilitated by installing a few external controls on the steering wheel, bypassing the need to touch the screen while driving. This would go hand in hand with the "Drive Mode", and the app could controlled by a couple of buttons on the steering wheel.

1.4. Research Conclusion

Designing a tailored solution for garbage truck drivers requires a deep understanding of their specific needs and challenges. One of the primary concerns for garbage truck drivers is the need to optimize their routes to minimize time spent on the road while also ensuring timely collection of garbage from every location on their route.

Additionally, drivers need to be able to quickly and easily locate and navigate to new pickups that may not have been previously planned. To address these challenges, a custom app could be designed to include features like real-time route optimization based on live traffic data, geofencing to automatically trigger notifications and route changes when a driver enters a specific area, and turn-by-turn directions with voice guidance to help drivers navigate to their next pickup location.

Another important consideration for a garbage truck tracking and navigation app is the need for robust safety features. Garbage trucks often operate in residential neighborhoods and other areas with heavy pedestrian and vehicular traffic, so the app should include safety features like speed limit warnings, pedestrian alerts, and collision detection.

To ensure the app meets the specific needs of garbage truck drivers, extensive user research and testing will be necessary. This may involve conducting surveys, interviews, and focus groups with drivers to understand their pain points and how the app can be designed to better serve their needs.

By taking a tailored approach to designing a tracking and navigation app for garbage truck drivers, it is possible to create a solution that is both highly effective and easy to use.
By incorporating features like route optimization, geofencing, turn-by-turn navigation, and safety alerts, the app can help drivers save time and operate more safely, ultimately leading to better outcomes for both drivers and the communities they serve.

2. Production

2.1. Scope and limits of design

Introduction

Implementing all the previously discussed features would be amazing. It is important to clearly define the target audience, so we keep the end-user in mind throughout the process.
However, as a bachelor student, I work with certain limitations. It is important to keep this project within the scope of my resources and technical knowledge. Within the time I have to design this project and based on the data I was able to collect, it also makes sense to impose a geographical limit.

Geographical area

Each environment has its specific challenges to navigate in terms of traffic rules, infrastructure, and geographical features. Because of my research in the area of Ronse, it makes sense to limit this project to the area of Ronse. The focus is on the routes in the city center, more specifically the Blue Route, otherwise known as “Lisa’s Trail”. The streets that make up Lisa’s Trail will be used as a sample for the design of my product.

Technical aspects

As a third-year bachelor student, I must acknowledge that my knowledge and technological resources are limited. Therefore, there are certain constraints that I need to consider before developing the navigation system for garbage truck drivers.

Furthermore, I have limitations in terms of the data that I can work with and a restricted budget to acquire support technologies that may be required.

Additionally, as I will be developing this system on my personal 2017 MacBook Pro, I need to keep in mind that there may be constraints in processing and computational power. Despite these limitations, I am determined to create a functional and useful navigation system that meets the needs of garbage truck drivers.

Target audience

The end-users, as such the target-audience for BINAV is garbage truck drivers.
Truck driver profiler is, multicultural, multilingual. Also note the use-while-driving concept: short, frequent stops allow for short, frequent interactions. A simple and clear interface is essential.

2.2. Implementation

A. Introduction

All source code for the development can be found in the Github repository.
Each section using code will also have a separate header “Github: /source/” with a link. This will direct you to the Github repository, with the code used for that specific subject.

There are a couple of **”.html”** files which can ether be directly opened it the Browser of will require a live server (Live Server - Visual Studio Marketplace, n.d.).

B. Dataset

As the goal initially is to digitalize a map, we need data that represent street names which form a route.
First of all; I need the set of streets that make up the route.
Renewi Ronse was unwilling to share their routes or custom maps. But luckily I was by a paper magazine. The yearly Municipal Magazine (IVLA) happened to contain a list of all the streets in the center of Ronse,grouped into garbage pickup routes.
This was a very good starting point. Only ‘slight’ issue here is that the streets are listed in alphabetical order, and not in the order they need to be driven.

Now that I had the street names of the route in the center of Ronse, I somehow need to create their optimal order for efficient route.
In other words, to start with, I need coordinates.

Research

Using Open Street Map Data

After a long quest for developer friendly data sources, I discovered Open Street Map (OSM).
OSM is the largest available open source geographical dataset. It additionally has other datasets like a map of GPS locations around the world.

The hard part was using their metric, as OSM creates their own IDs.
For example, the Veldstraat in Gent has ID ‘762619506’ [1]: .
To complicate this even more, each intersection of each street is represented by a node with its own ID. I can not simply add a street into my dataset, but need to manually add every one of its nodes. After having inputted all this, OSM creates a route for you.

This leads to another issue: You cannot limit the route to a geographical area.
As it turns out that each node ID potentially represents multiple nodes, across the world.
As such, the first OSM routing for Lisa’s Trail included stops in South Africa, the Netherlands, and even Paris.

Using Turbo Overpass

I used (Overpass Turbo, n.d.) to understand the data as it allows you to query all OSM data and visualize this neatly on their map. But more importantly it also provided the coordinates of the query json format.

1 https://www.openstreetmap.org/way/762619506

But finding the right query is not easy as the provided documentation (Overpass Turbo/Examples - OpenStreetMap Wiki, n.d.) has a steep learning curve, and is very succinct.

Programming & testing

I created a couple of lightweight scripts for testing queries and data.
Just to note; these are only the successful ones.

Version 1: Iterating

Github: *Basic-implimentation/2-basic-features/1-data/1.1-get-iterateStreets.py*

The first successful Python script had the purpose to query data by using the "overpass" package [2]. and output this into JSON.
Then I iterated the original array of streets of Ronse and requested all nodes for each of those streets. This was rather a slow process.

Version 2: All Streets

Github: *2-basic-features/1-data/1.2-get-all-streets.py*

To speed up the process, I decided to query all streets at once. The first version of this script yielded successful results (see the Github commit history) but I quickly ran into a problem.

PROBLEM: Query

If I would query a street, just using a raw list of it's coordinates, finding an optimal route would not be efficient for the following reason:
If you go from street A to B; there might be a side street, C, leading back to B. So It mighty be more optimal to first go to C, sot the order would be A, C, B.

I had to take all connection points between streets into consideration. This made the query a lot more complicated. As I needed to query not only each street, but rather each connection point to and from each street.

I found a lot of useful help at Stack Overflow (Get Adjacent Ways Overpass API, n.d.), and made changes to the query resulting in the data of the route "Lisa's Trail", which currently being used throughout the project.
The resulting data can be found in the data folder [3].

Version 3: Folium

Github: *Basic-implimentation/2-basic-features/1-data/2.3-create-foli.py*

Now that I had the data, I looked into displaying the data on a map. I found a brilliant guide (Khamis, 2021) for using OSM data in Python and also displaying this on a map.
This also led me to rewrite some code of previous versions.

Now I could converts the gathered JSON data directly into a interactive Leaflet map on a newly created HTML page using the "folium" (Folium, 2022) package. One of the reasons I used Leaflet is, that it's an open source library that can also be used in React (Introduction | React Leaflet, n.d.) to display OSM maps; in other words exactly what I was looking for.

---

2 Python Overpass, n.d
3 https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation/data

PROBLEM: Coördination systems

The map pointing to the coordinates of the data, was empty.
Searching the entire world, and finally found the streets back in bright and shiny colors in the middle of the Arabic Sea, east to the Horn of Africa.

Apparently Overpass uses coordinates in the order of [longitude, latitude], while Leaflet uses [latitude, longitude].
So if you find yourself in the middle of that ocean wondering why you're home made GPS directed you that way, just switch the coordinates around.

After switching around the coordinates we now had a HTML page with a highly customizable, interactive map, which displaying the route of Lisa's Trail.

C. Basic Features

Here we will go over the more technical evolution and code and some of the problems that came along. I will cover a couple of technical highlights of each step, and keep in mind that this still only describing the end results of the process.
With this Leaflet template we already visualized all data, but there was still most of the work to be done. I added some functions and made a Marker walk the streets in a somewhat random order.
Visually there is still a lot to be done, but can work as a sandbox to do rapid experiments.

Progress Tracking

Version 1: basic

Github: Basic-implimentation/2-basic-features/1-progress-tracking/1.1-progress.html

One of the first problems I wanted to tackle was progress tracking.
So I made a new script that was able to track how much of each street was done, as my concept was that the app will use the trucks location to mark if a street is finished or not.
(street names are replaces with colors for testing purposes)
There where still a lot of problems and things that needled to be fixed (described in more detail in the second version).

Version 2: smooth

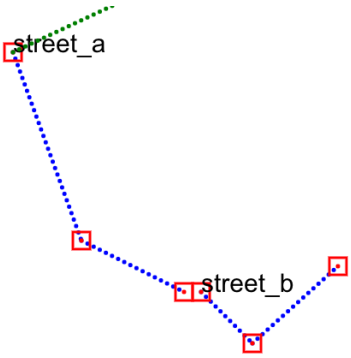
If you want to calculate how much of the road is finished and the road only consists of 2 nodes, the percentage will go from 0 to 100. So I started to think, and came upon the idea of adding multiple points in between, that would make up a street.

PROBLEM: Intermediate points

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/1-progress-tracking/smoothness.html

To make this smoother I used functions out of older projects of mine (https://github.com/jonatan-verstraete/js-canvas/blob/main/setup.js) where I experimented with animations with the Js

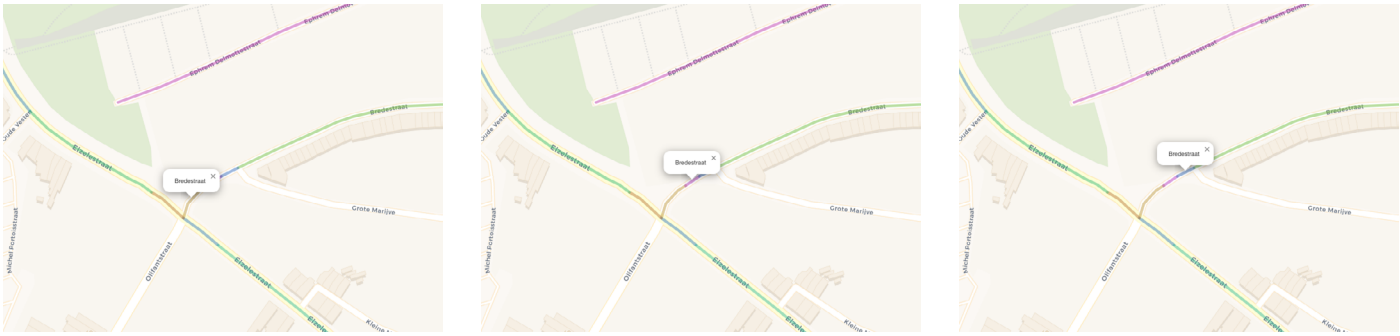
canvas Api. These functions plot points in between every node every 8 steps, resulting in the following:



Now the percentage will be a more smooth transition and is just: checked-nodes / total-nodes. This resulted in the following:
Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/1-progress-tracking/1.2-progress.html

PROBLEM: in between

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation1-data/1.3-update-data-conversion.py



[Bredestraat - part 1] *[Bredestraat - part 2]* *[Bredestraat - part 3]*

Another problem was that each street was made out of multiple parts which bared the same name. Adding all waypoints together would create a huge mess of overlapping lines, so I created another script to filter the json data yet again.

Now each street is clearly made out of parts with an ID and each part exists out of a couple of nodes.

Grid

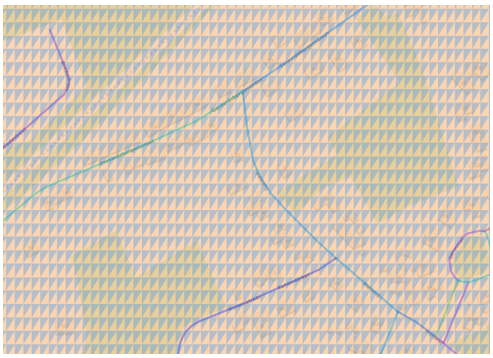
Concept

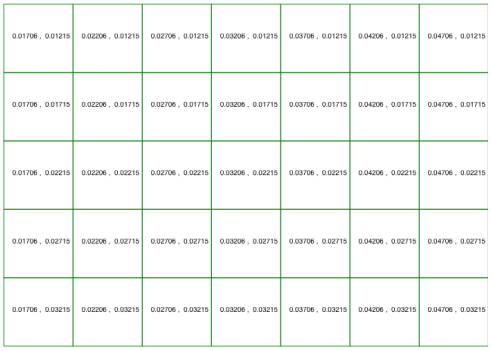
Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/2-grid/2.0-grid.html

Another, new, problem arises when using real time location data. E.g. If the truck is in between two streets, we can't just check the nodes of one street, because we don't really know which street the truck is at.

So there will be need for a system that is verify the area around the truck.
But then this comes with a performance issue, because we would need to check all the coordinates in a certain area every n-steps. Depending on the precision, a coordinate system is to 1.11 meters (Decimal Degrees - GIS Wiki | the GIS Encyclopedia, n.d.) distance which would result in a lot of iterations.

Mapping systems used in Terravision (Terravision, n.d.) or more modern system like Google maps or terrain generators, all use systems to divide everything in a grid. I used this idea to create a grid like in the images of test phases below.
Now it was time to assign each street node to a node on the grid.
Then we can search in a certain radius around the truck's position, iterating the grid by the detail of that grid.





[triangular view of tiles] from 2.0-grid.html *[visualized grid] from 2.1-grid.html*

Version 1: tiles

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/2-grid/2.1-grid.html

However implementing this grid method in the current map, was very messy and the concept completely backfired.

Performance became the main issue. It took 7826.071 ms to set up the map. Iterating all points in an area also took very long as calculating the distance to a point uses computationally heavy functions like "Math.hypot", which calculated the Euclidean norm (the square root of the sum of squares of its arguments).
My strategy was to have complete control over every coordinate that if need be, coordinates could be added to the route which where not necessarily in the original area.
But it could not handle positions out of the route it's area as that would require creating another grid which ended up with dire performance.
I thought about saving all the gathered data in another JSON file, but this might rather complicate things in the long run so I stopped coding and continued to research.

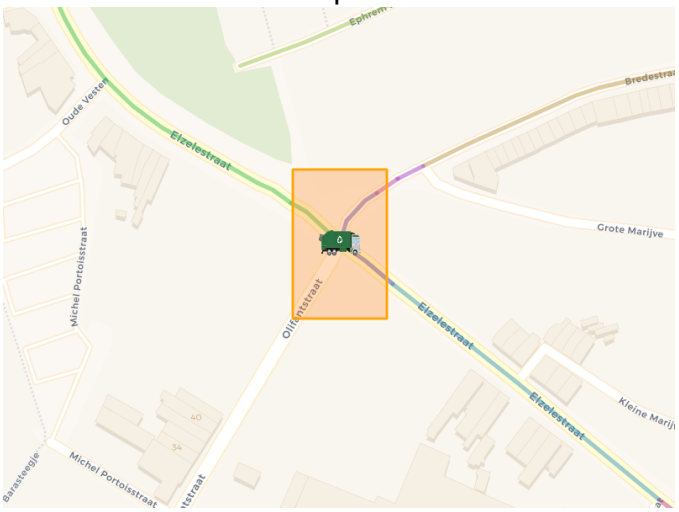
Version 2: uniform

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/2-grid/2.2-grid.html

I decided to use concept of a uniform grid as my current research pointed out that this method is rather common. The concept is rather easy: you divide the terrain or map into equal sized blocks. Each of the chunks (or commonly buckets), will contain a list or array of objects which reside in them.

But dividing everything up in a grid would result in a similar outcome as the past attempt, because checking in which tile the truck is currently at, would require calculating distances again, in order to verify that you are not in another tile (bad performance).
Another problem with using tiles is a scenario where the drivers gets a request to finish a street in another village, then we would also need to recalculate. So using a fixed grid is only good if you know the exact boundaries.

In order to prevent all this, I reformed the concept so that it only scans a tile surrounding the truck as can bee seen in the picture below:



[Visualization of the area surrounding the truck]

This resulted in a huge performance boost.

console.time, individual functions:

Setup (reate the grid, assign all street nodes…)	112.260 ms
scan the a single tile in the grid.	6.363 ms

One bad aspect was a lack of control over the grid: the coordinates of the grid are somehow a bit weird numbers, to generate the square on the above image I used the following, very messy function in order to translate the grid coordinates to real-world coordinates:
`const getX =(a)=> +String(a).split('').splice(3,50).join('')/100000 + 50`

I was content to a certain extend, a pity for the somewhat weird number and rectangular scanning area, but a relatively good performance so I couldn't complain.

Version 3

`Github:` https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/2-grid/2.3-grid.html

After spending a lot of time on these grids and functionalities, I finally had a small "Eureka" moment:
I though that it might be more easy to simply round the coordinates off. Coordinates in this system use 6 digits after the comma, which results in an accuracy of 1 meter.
Positions rounded to 5 digits would result in an accuracy of approximately 10 meter.

Positions that are within this 10 meter of each other would now have the same coordinates, so you could put them in a shared dictionary or map, with the rounded coordinate as key.

This would resulted in a similar effect as the tiles but it uses the grid system of the coordinates themselves. Not only that, but with the benefit that there is no need to create a grid or to scan the area surrounding the truck.

Therefore this would not only resolve all previous issues, but also boost performance, add control and increase scalability.
In case of adding a route outside of the original route, this system does not require to recreating anything; simply add the rounded new coordinates to the list.

Performance test of an average of the whole route:

console.time (20 iterations), comparison:

Version 1	7826 ms
Version 2	3633 ms
Version 3 (current)	32 ms

Coming from nearly 8000 ms, the current version is about 250 times faster and can probably still be improved on a lot.

Transferring parts

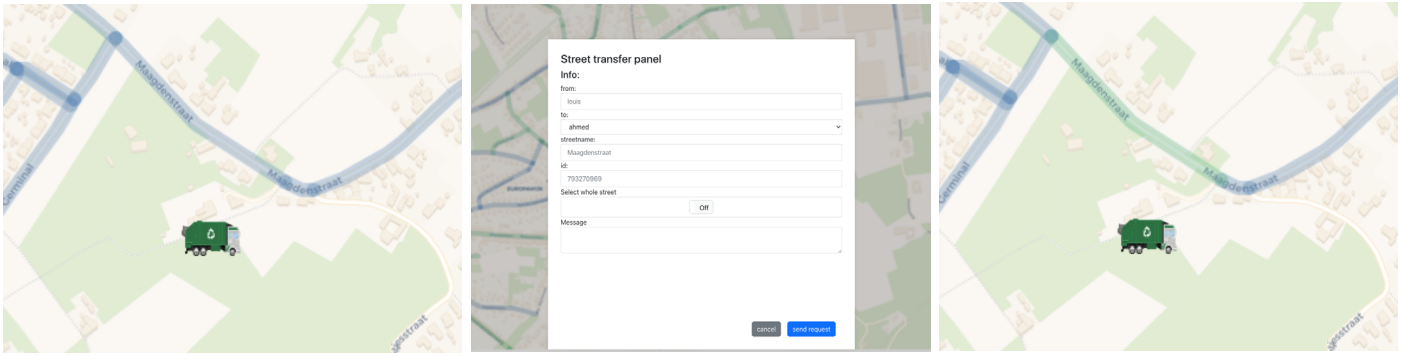
Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/2-grid/2.3-grid.html

The goal here was to be able to transfer a road, or part of a road to another user.
This is not very hard to do, but if done well requires a complete review of the current ecosystem. Basically everything needed to be updated to be as efficient as possible.

In order to transfer roads in a more realistic scenario, I started to work on something that would handle these requests from diver A to driver B.

To simulate this as a lightweight example, without using the likes of Node Js, I create a sort of fake server object (in future versions this could be an Api, which enables the user to download routes and handles the traffic between users).
One of the features it has, it that a driver does not have a database of the roads, and that all data is passed on from this Server Object to the drivers.
This way the driver's device does not need to make any heavy calculation and/-or configurations, and only needs to download or receive the right data.

The basic idea is to user-A get a popup form by clicking on a road (or part of), which gives him the option to select a user and an option to select the whole street.



[before transfer] *[prompt transfer after click]* *[after transfer]*

This form submits the id of the street to the Server object. As this server has all raw data of all streets, so by receiving this id from user A it can retrieve the raw street, which it sends to user-B. Then user B get a popup form which results in sending a response (accepted or declined) via the Server back to user-A.

PROBLEM: response

When a road is transferred is basically is deleted at user-A, and recreated from scratch at user-B. One challenge was that user A needs to wait for a response from user A, but I don't find it optimal that user-A is waiting for user-B's response.
So I created an object holding all pending actions. For user A the action has the original ID of the street as a key and a function to delete the street from its memory as value.
This is also more efficient for user-A, as the action is just stored and is only retrieved (and deleted) on the response of user-B.

Offline

Concept

The reason I want to make the map offline, is not only for performance reasons, but also for better security and privacy and overall less costs of internet traffic.
But the reason that instigated the idea to go offline is that in more remote areas or stormy weather a wireless device its connection might be less reliable. So if you depend on a digital map, you need all time access.
In this case, we want the truck to be as autonomous as possible, so the purpose is that a map can be downloaded from the Api/server. As such, like Maps Me, the goal here is to make a driver able to download a route.

A map is made out of individual images, each one arranges in such a way to give the illusion that it is one larger image. This illusion is broken, by the visible loading if those images.
Even Google Maps has this, when you zooming, it gives a white grid off where the images should be, as they are not loaded yet and only starts rendering them once you stop zooming. But once the images are cached in memory, the zooming goes without hick-ups. The goals is to make this hiccup as small as possible.

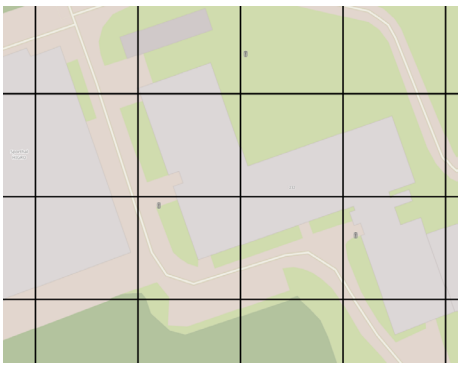
If we take the two images below from the Open Street Map (https://www.openstreetmap.org/#map=20/51.08733/3.66873&layers=H). The first is zoomed on 20 (maximum) and the second

on 19. Visually not a lot has changed, but actually all the images of ‘zoom 20’ have been replaced by images from ‘zoom-19’.

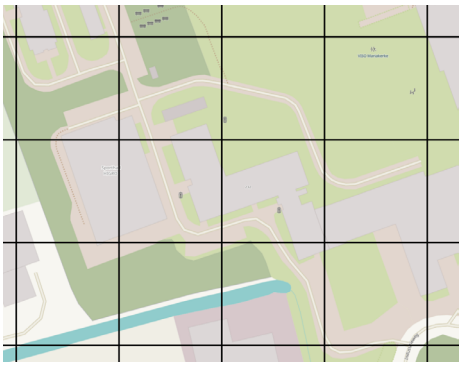
[zoom 20]

[zoom 19]

But if we add a simple CSS border around the images, we can clearly see that the zoom determines the images to load and that the size and amount of images stays relatively the same. The only difference is the amount of detail in the images (for performance reasons), as street names and icons are hidden with a when zoomed out.



[zoom 20, with css border]



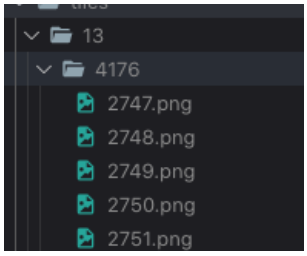
[zoom 19, with css border]

Offline Map

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/1-data/3.0-download-area-offline.py

With the lightweight “shutil” and “urllib3” Python library’s you can download any image, so these map tiles as well.
This script allows me to download any area, to any zoom. The zoom factor is very important here, as each higher zoom level results in an exponential amount of images.

A seen in the following image, this folder contains nearly 6000 images (bash tree | wc -l).
It is the map of the route (a parts of Ronse) and doesn’t take long to download and is in total about 52mb.



Larger images are about 25kb (zoom 13) and smaller images can go down to 1kb (zoom 17) in size.

PROBLEM: Legal aspects of bulk downloading

As I did more research I found that bulk downloading is not allowed (https://operations.osmfoundation.org/policies/tiles/).
And downloading more than 250 tiles or a zoom of more than 13 is also not allowed. So I had to figure out another way.

Solution 1: tile server

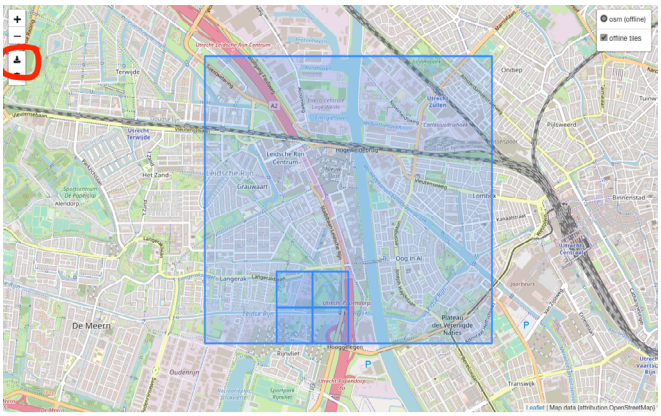
This article (Gibbard, n.d.) helped me a lot. The reason just downloading these maps is not allowed is because OSM is a none-profit organization and they a have limited bandwidth on their self-paid servers.
His solution is to download a larger area and then download the necessary images from you a self made, image server.
But this is a rather time expensive solution. You also would need a very heavy computer and this seems like a project on it's own.

Solution 2: Osm Lab

After doing more research I found Osm Lab which allows to download tiles by country (OSM QA Tiles Country Extracts, n.d.), and Belgium was only 500mb. But the main problem was that it only supports 12 zoom, which does not include detailed street names, so is not usable for this project.

Solution 3: idb package

Then I found the 'idb' NPM package, which allows you to store these tiles/images on the fly.
As we can see in the following example from https://allartk.github.io/leaflet.offline/:



[example page] from https://allartk.github.io/leaflet.offline/

Is becomes very easy to download tiles, and even gives the user the option to use offline or online tiles.

I did some testing, and found that a downside is that the offline mechanism it is not automated and can be buggy.

End solution

`Github:` https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/4-offline/4.1-offline.html

Eventually I just downloaded some parts over a longer time (2 days), with a timeout, so It definitely could not be considered bulk downloading.
But if this would be build into a large scale application the developer should contact the OSM Foundation or find alternative solutions like Osm Lab.

I choose for a zoom level from 15 to 17 as I figured these would be the most essential.
A zoom level of 17 is the maximum zoom that still gives all essential details and street names.
If we would use a zoom-level of 19 (maximum), which would give a lot more detail, but which might not be as essential, like trees. But also for performance reasons, as a zoom-level of 19 would take 64 times as much tiles as a zoom-level of 17 (Zoom Levels - OpenStreetMap Wiki, n.d.).

D. Creating a route

Introduction

Now that we have most of the functionalities for the map, it's time to look at one of the most essential things, acquiring the correct route as currently the streets are in a somewhat random order.

Concept & Research

To find route, we can use path finding algorithms (Atuhaire, 2022) to search for the most optimal oder of roads.
The most common routing algorithms, like the famous Dijkstra's (Dijkstra's algorithm, 2023), are primarily made to go from points A to B.
But having more than 800 waypoints of destinations requires a different approach, as basic, often brute force algorithms would not be time efficient at all.
One of the most renounced is the TSP or Traveling Salesman Problem (Wikipedia - Travelling salesman problem, 2023). The goal is finding the shortest, yet most efficient route for a set of destinations.

There are many, more of alternatives to the TSP, like Orienteering Problem (Mitchell, 2000) or Dial-a-Ride Problem (Cordeau & Laporte, 2007), but it becomes very complicated very easily, so in order to not fall into to many rabbit holes I decided to stick with TSP.

However, the standard TSP only visits every location only once and ends at it's starting point. But as garbage truck's can visit the same location multiple times, we have to the Multiple Traveling Salesman Problem or MVTSP (NEOS Guide, 2022) variant, which using the same destination multiple times.

Usually each algorithm also has multiple variations making use of various techniques like, dynamic programming (breaking down a problem into smaller subproblems and solving each subproblem (Bmkmanoj, n.d.)) or heuristic approach (Derecia & Karabekmez, 2022).
That said, there are a lot of ways to solve our problem of finding the shortest route between the streets of the center of Ronse, so I immediately tested some out. I choose 3 standard algorithms to test: Brute force, Two-opt and the ACO (more details later):

Algorithms

Brute force

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/5-route/1-TSP+MAZE.py

The maybe simplest method of solving the TSP by systematically trying all possible solutions until the correct one is found. This is done by calculating the distance between all possible permutations (possible combination) of destinations and selecting the shortest path (GeeksforGeeks - Brute Force, 2021).
Generally brute force is considered slow and not scalable.

Allthough there a lot of waypoints, I tested it to compare and see the possibilities. But as I kept waiting and waiting, I had to quit the program. I figured this was probably due to the amount of waypoints.

PROBLEM: improve performance

Github: Basic-implimentation/2-basic-features/5-route/helpers.py

I figured that I might not need every waypoints to figure out the shortest route, so I made some functions that only use the first node of each part of a street. This reduced the 800 to a 100 waypoints.

Additionally, I once programmed maze solving algorithms (https://github.com/jonatan-verstraete/js-canvas/tree/main/maze), and I remember being surprised how fast they run. It's also not hard to imagine streets as a maze, and this might be a lot faster, so tried to combined it with the brute force.
But then I calculated out that 10 waypoints, would take 3628800 permutation (not even calculating anything). And 100 waypoints came to a number with about 151 zero's.

As such, I never got to test it completely as this might take more time than the current age of the universe (If a Supercomputer Can Test 100,000 Trillion Combinations per Second, How Long Would It Take to Test a Vigintillion Combinations?, n.d.).

2-Opt

Github: Basic-implimentation/2-basic-features/5-route/2-two-opt.py

This is a local search algorithm that works by swapping two edges in a tour, provided that the swap results in a shorter overall distance. This process is repeated until no further improvements can be made. In the context of TSP, two-opt is used to improve the solution obtained from another algorithm. (Source: Wikipedia - 2-opt).

Generally it works well for very big problems, but can be slow and not optimal with not enough time (Lathrop, 2020).
As I had become weary about the time it could take, even with a 100 waypoints I gave a limit of iterations.

Code & testing

The first results where not brilliant, but also did not become any better, it was nevertheless a lot

faster (to test if the result was successful ,I had to reserve the coordinates to not end up at the Horn of Africa, again)!

[Route before applying 2-opt] *[1000 iterations]* *[10000 iterations]*

A 1000 iterations nearly took 2h's so I had to run the other version over night.
However the algorithm had made some weird choices and for some reason left out a lot of streets. I figured this was due to the way the algorithm works so thought it might be good to test out the next algorithm of the list.

Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) is part of the family of Generic Algorithms.
A GA is an adaptive heuristic (a heuristic solution that changes to be more optimal) algorithm (GeekforGeeks, 2022). It works by randomly populating solutions which evolve, and the ones surviving a set of rules, generate the next generation and in doing so find a better solution.

There are multiple GA's, but the one I was particularly interested in was the ACO, this also due to personal interest, as I once tried program an ant colony myself (https://github.com/jonatan-verstraete/js-canvas/tree/main/games%26simulations/ants).

The ACO works by simulating the behavior of ants as they search for food. The ants start at the depot, or HQ, where the garbage truck in question would be parked, and they move along the graph by choosing the edges with the highest pheromone level. Pheromones are chemical trails that ants leave behind them, and those trails serve as a form of communication which allows ants to find the path to a food source.
After each iterations, the path becomes better and in doing so the solution as well.

Code & testing

`Github: Basic-implimentation/2-basic-features/5-route/3-ACO.py`

It took a while to make the code work, but found a lot of help in online sources as there is a lot of research available on this subjects.

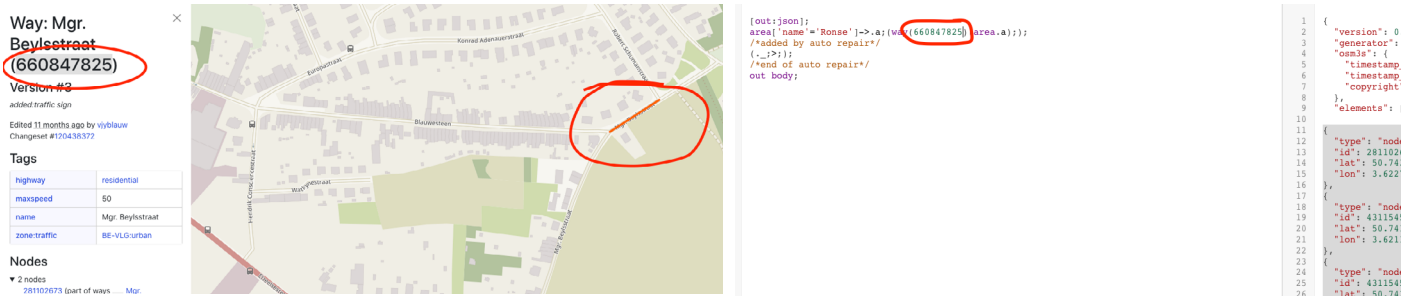
The first successful test, although far from perfect looked promising:

[ACO, 500 iteration] screenshot from www. geojson.io

PROBLEM: Reforming the data

But first test also gave some weird results, similar to the two-opt version.
So I investigated the code, and found that the data being used, had some issues. Some streets where not connected to the whole and some streets had missing parts a seen in the following image.

So I manually added missing streets using Tubo Overpass (Overpass Turbo, n.d.) and the OpenStreetMap (GmbH, 2022).



[Open Street Map, identifying id of street] *[Over pass turbo, requesting data]*

For the next run, I used about 400 waypoints, taking the fist and the last endpoints of each street part.
As this would take a while I let it run over night.
After a failed attempt because I used the wrong dataset, I finally got a relatively good looking route:

[overnight run]

However, there are still a lot of weird choices made by the algorithm or the digital ants.
These choices could be due to the algorithm's code, the arguments of the algorithm (e.g. the number of iterations), or the dataset.
I did another few tests but with very similar to near-identical results.

Finalizing the Route

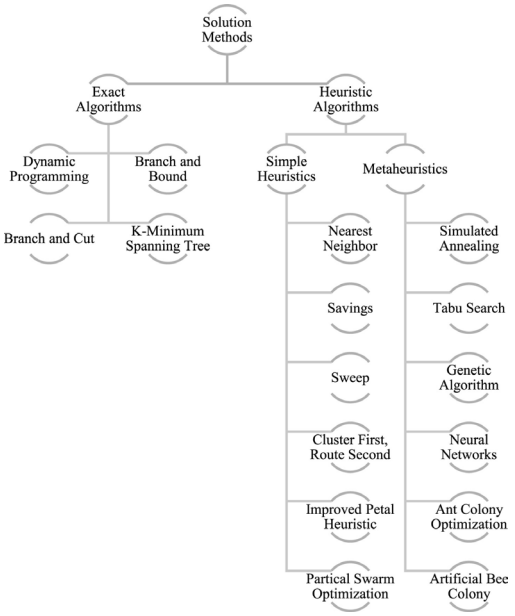
`Github:` [`Basic-implimentation/data/Routes/route-ronse-center.json`]

The result of the ACO was relively good, and if you look closely, the underlying structure of a route is present. So I took a more heuristic approach again, and used geojson.io to manually filter out the weird choices. This gave a surprisingly good result.

[Screenshot of edited ACO results] from www.geojson.io

Final notes on these algorithms

I found a paper (Derecia & Karabekmez, 2022) trying to effectively route garbage trucks in turkey. This gave me great insight in the usage of possible algorithms in a real life scenario.
As the paper states, in a real life scenario or enterprise solution, there is no single solution or single algorithm, it's a combination of a lot of factors, algorithms, mechanism, data analysis, deep learning systems, as can seen in the following diagram, taking an approach to solving this kind of problem.



[Algorithms diagram] from www.sciencedirect.com/science/article/pii/S2772662222000480

So if this project ever would go in production, it is important to note, that using a single algorithm is probably not the right approach. Using a combination of multiple techniques in the field of routing and path-finding are required to give reliable and optimal results, but this is out of scope for this specific project.

E. Routing

Now having a map it is important to navigate the user in this route.
`Github:` https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation2-basic-features/6-routing/

PROBLEM: Usage of existing routing software

There a re a lot of libraries and frameworks out there but none of them offer exactly what I needed or are not free to use.

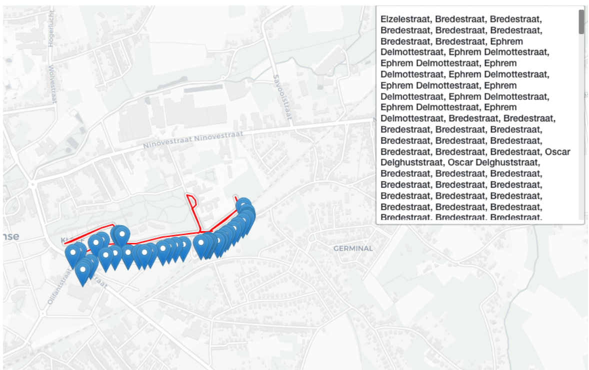
Most routing software, like Google Maps, in made to navigate a user from point A to B.
But a garbage truck goes from A to B to C and back to B, multiple times.

[Possible route of a garbage truck]

Not only this, but most software does not allow a lot of waypoints or destinations, so looked for software that did (17 Best Free Route Planners With Unlimited Stops, 2023).
Google Maps only allows 10, destinations, although there is a known workaround (Maskell, 2022), it is nor developer no user friendly and impractical for this project.
Mapbox allows a total of 25 destination (Directions API Playground | Developer Playgrounds, n.d.).

Badger (17 Best Free Route Planners With Unlimited Stops, 2023) claims to allow unlimited waypoints, but is not free of use, and with good reason. As calculating waypoints is computationally expensive and usage of resources, like computing time, grows exponentially with the amount waypoints.

Leaflet's standard Routing Machine
Leaflet's standard Routing Machine (Liedman, 2015), being created Leaflet seemed promising, even though I could not find any references on the amount of waypoints.
So I did some testing and found that the limit was more than 50 an even went up to about 130 waypoints.



[50 waypoints, does work] *[500 waypoint. Does not work]*

Gray area solution

As we have about 500 waypoints, I thought it might be an idea to split up a route in sections, for example, 5 sections of 100 waypoints. Although this theoretically would work and would be a nice asset to this project, it is not usable is a real life scenario for the following reason:
Currently a truck follows the exact route, but what if there are roadworks, and his route need to be re routed?
In an optimal scenario this route would be completely recalculated, not only the single destination.
So splitting a route in 5 parts might hinder a path finding algorithm to find a new optimal route.
Therefor I would consider it rather a waste of time in the long run, to implement this, as this technique would never be used in a production version.

Code from the screenshot and routing can still be found in the App (https://github.com/Jonatanfroeling-user/Bap-garbagetruck/blob/main/App/src/js/components/Map/Routing.jsx).

F. Real Time Location

Concept

The basic functionalities are all done, and it's time to look at one of the key aspects of this project: real time location sharing of the truck.

One of the first ideas I had about this projects before I even started it, was to use a Raspberry Pi as a dummy truck driver which you can move on a normal sized physical map to simulate a truck driver.

The goal was to use a Raspberry Pi (4 B) with a sense-hat for this, because I do not have a gps tracking module and WIFI/indoor location tracking is not accurate enough.
The idea was, that when the sense-hat moves, it detects movement, and adds the change in movement (detected from it's accelerometer) from the previous movement.
This change in movement we then add to the current location which should simulate the truck's movement in an accurate way. And by multiplying or dividing the movement we could enlarge the movement to move in meters in the coordinate system.

Research - WebSockets

This meant coing back to the drawing board.
In most school projects we learned to use http and how to set up a REST api. But in the course Web Of Things I learned the basics of MQTT (paho-mqtt) and indirectly or accidentally also learned about the WebSocket technology, i.e. socket.io.
Some described socket.io simply as a layer on top of the standard WebSocket or "ws" library (rsp, 2016), and that it actually slows down the process as it is made to be compatible with all browsers and versions, as it adds another calculating factor to the performance.

But it made sense to me that WebSockets have generally better performance than http, considering real-time data as they keep a direct connect or a bidirectional connection.
My suspicions where confirmed by a performance study (WebSocket vs HTTP Calls - Performance Study, 2019), which stated that WebSockets are expected to be about 5-7 times faster than http.

WebSockets are considered the next generation method of asynchronous communication from client to serve (Walsh, 2014), so for this project I wanted to use this method not only for performance reasons, but also as an interesting learning path for future development.

Coding & Testing

Version 1 - Accelerometer

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation/7-realtime-location/Accelerometer/

PROBLEM: sense hat - communication

One of the problems is that the sense-hat cannot directly communicate to the front-end, as JavaScript does not have direct access to the sense-hat. So in the end, after long discussions with ChatGPT, I ran a Flask-Socket.io server on the Raspberry PI which emitted data to the front-end on a MacBook which displayed this data in a front-end.

Problem: sense hat - movement

Finally getting the whole thing working I found that the marker on the map (which moves by the received movement from the PI), made unpredictable and rather random movements.
But after rewriting the code and trying different angles, still the same random like movements appeared.
Most movement was given by tilting the PI and not by moving it in 2D directions.

So I started to investigate and debug and borrowed a, in perfect order, Sense Hat from Artevelde, but it gave the exact same readings.

Ultimately I reached out to one of my teacher, Frederick Roegiers. He immediately pointed out that, as I by then feared, I misunderstood the concept of an accelerometer:

As it turned out, and accelerometer can only detect statical movement,
not dynamic movement. So I had to look for an alternative:

Version 2 - Using the Route

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation/7-realtime-location/Route/main.py

As I already spend a lot of time on the previous attempts, I took a step back.
As “real” real-time locations where already not an option anymore, I realized that I could use we the previously created route data as source for real time locations.
By simply adding intermediate points into the route (as done previously in progress tracking), we can get relatively realistic movement for e.g. a garbage truck.

PROBLEM: progress

Github: https://github.com/Jonatanfroeling-user/Bap-garbagetruck/tree/main/Basic-implimentation/7-realtime-location/Route/test/

However after testing the route only about 80% of the streets their progress was done.
I changed the accuracy from 0.0001 to 0.00001 which now checked the nodes in a 10meters. But the results where still not good.

After a long time staring at code and debugging it I found out it was the route-data itself. Although the intermediate points where added in between A to B, it was still strait line from A to B, the map and progress tracking however where less linear as they used a lot more waypoints. (image of map)

This was solved by manually moving and adding points in between, using, Turbo Overpass, OpenStreetMap and geojson.io and a lot of converting between the [longitude,latitude] and [latitude, longitude] order.

2.3. External Control

A. Steering Wheel Control

Introduction

During a conversation on the requirements of this project with my mentor, Dieter de Weirdt, he suggested to conceptualize an IOT device for the steering wheel of the driver. This device would ease the user experience as it would allow the driver to control the app, with a couple of buttons.

Concept

The idea that the end-user can easily use the app while driving is paramount within the product design of this project.
As such, when looking at more modern cars and trucks, most of them incorporate buttons on the wheel. Indeed, the car-manufacturing industry has done the diligence of researching this far better than I ever could.

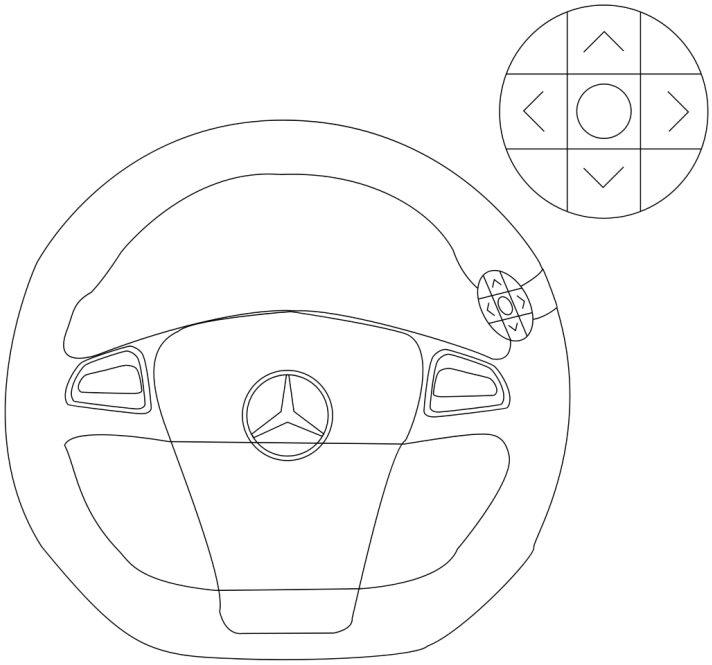
Therefore, I found Mr de Weirdt his conceptualization, a solution I had been overlooking for quite some time:

Initial conceptualization

After conducting some research on steering wheels and speaking with a few truck drivers about their experiences on the road, it has become clear that many individuals feel their cars are equipped with too many buttons on the steering wheel. While some find these features useful, many find them to be distracting while driving. In fact, most drivers tend to have a few go-to buttons that they frequently use, but rarely if ever use all the buttons provided by the car manufacturer.

Considering this feedback, it becomes clear that if we want to ensure safe and effective interaction between truck drivers and our app, we need to limit the number of buttons required to operate the app. This was taken into account during the development process, with a focus on creating an interface that requires the fewest possible buttons while still providing all necessary functionality. Ultimately, our goal is to ensure that truck drivers can use our app in a safe and efficient manner, without becoming distracted or overwhelmed by unnecessary buttons and features on their steering wheel.

Product-design



Description

An easy to use, click-on interface, that works on all industry standard driving-wheel models. Which can be connected to the BiNav App through wireless connection.

Features

- Left, right, up and down navigation buttons
- Central OK button
- Click on
- Removable
- Battery powered
- Lightweight

Conclusion

In conclusion, the development of an IOT device for the steering wheel of a truck driver has been conceptualized with the aim of improving user experience while ensuring safety on the road. The focus of the design has been on providing all necessary functionality with the fewest possible buttons to avoid distracting the driver while driving.

By incorporating this device, truck drivers can easily interact with the app while driving, making the overall experience more efficient and effective. It is hoped that the implementation of this concept will not only improve the user experience but also enhance the overall safety of driving on the road.

3. Conclusion

Introduction

Here we will discuss the final results of the research and offers insight into what the findings imply for the field of waste management and the development of garbage truck app
As we saw there is room for a lot of improvements, specifically in the fields of communication, maps. An app could provide a possible solution and I believe this would be of great merit to the drivers of garbage trucks that lack these solutions.

Potential future of the app

Overall I would love to see the finalized version of this project and be actively used by garbage truck drivers. However, if this would go into production, but this would be to large of a project for me alone and would rather be put forward to an enterprising company looking for such a solution, like Renewi.

It would be a large scale project, that would require multiple talents covering multiple field and technologies like routing, route optimization, data analysis, integration of telematics, designers... etc.

Currently the focus, in the product-design, as well as technology lies on the end-user, the driver. But these technologies, like routing algorithms are still rapidly changing (Derecia & Karabekmez, 2022) and the app should be very scalable to a possible fully automated future.
And in this possible near future, drivers may be replaced by automation through AI a.o.
Yet even then the app could remain relevant as it uses the same features like progress tracking, and route optimization which will still be needed.

This can be achieved by shifting the focus from the end-users experience toward these technologies and their integrations.
A user-interface might only be applicable to the administration as they might require the same interactive map of the driver, with a lot more options.

Self reflection

For me this was the biggest project I ever undertook. It took me completely out of my comfort zone in terms of scale, knowledge base, as problem solving.
I had to climb many steep learning curves, on a variate of new subject like algorithms for route optimization, OSM data. Not everything I reserached I used in the endresult, like Webscokets and the Raspberry Pi Sense Hat, but it did learn me a lot of things I know I will be usefull in the future.

General summary

A prototype routing app for garbage truck drivers was successfully developed, including custom features and a proposal for an external hardware add-on.
Overall, this project contributes to the field of routing technology by creating an option to optimize routes from a given set of streets. This thesis also highlights the role of technology in solving real-world problems.

BiNav is currently a prototype, but if it were to be actually used and developed, lots could be done. For instance, a record of the progress tracking logs could be analyzed to optimize routes, the dataset could be saved in case of a company merger or sell-out, and features such as transferring routes or drive mode could be developed into more detail.

Developing this app has been challenging, but rewarding.
During this project, I had to solve a lot of tricky technical and logistical problems by coming up with creative solutions. This project has taught me the importance of perseverance, attention to detail, and the ability to consult with peers. Additionally, I have gained insight into the technical aspects of app development and the importance of iterative testing. Overall, this project has been a valuable learning experience, and I look forward to using these skills in the future.

Attachments

Interviews

The interviews proved to be an important resource: they provided information that is not available online. Additionally, resources such as custom maps are considered precious and not easily shared. The interviews centered around several key topics:

- The driver's daily experiences
- Procedures for unexpected events, and how effective they are
- Common frustrations they experience on the job
- How confident they are operating phones, telemetrics, and communicating during unforeseen circumstances
- Opinions about my proposed app to streamline a driver's daily tasks.

Below is a short profile of each interviewee, and a summary of the main points they highlighted. I named the interviewees with their first name only to preserve a degree of privacy.

Alexander, 46 (not his real name):
Truck driver. Born in Moldavia, migrated here with his wife in 2002. His three kids were all born in Ronse. At home he speaks Romanian, but Alex is quite fluent in Dutch as well. Started as a garbage collector in 2002, graduated to garbage truck driver once he spoke enough Dutch to take the C/CE exam.
Frustrated by 'a job not well done'; such as care packages remaining unnoticed by crew, and stinking up the neighborhood. Frustrated this happens often, because he takes pride in his job and thinks this shows sloppy work ethic. Loves the convivial atmosphere with colleagues.

Frustrated by the multiple languages: "I come from Moldavia. I speak Russian, Romanian, English, Dutch. They speak French. Only French? What is this!" Throws hands in the air. Thinks the app is a great idea and I will become a millionaire. On the more realistic side, he specifically would love the integrated display feature, ridding him of all the 'garbage" screens in his driver cab. "No bells and whistles, flashing lights" he said, "keep it simple"

Hendrik VC, 35:
Born and raised in Ronse. After graduating highschool, Hendrik worked odd jobs for a while, then became a garbage collector at Renewi. After 8 years he took the C licence training and graduated to a driver. Has been driver for 2 years now. He loves 'his' route and knows it by heart, and when he is assigned different routes he hates the inconvenience of having to refer to a physical map while driving. He dislikes untrained student coworkers, although I suspect that is his little joke as him and I get along well enough to enjoy the occasional game of chess outside work hours.

Youssef K, 27:
Hates deskjobs, which prompted him to dropout of high school 5 years ago,
starting as a garbage collector a few months after. He was initially attracted by the higher wage. Youssef likes being at Renewi, but does not want to become a driver because he wants to remain physically active. I suspect his limited language proficiency is an obstacle for the driver training as well as the administrative part attached to the drivers job. Youssef has a mixed cultural heritage and grew up in Ronse as a native Arabic and French speaker. Due to my limited knowledge of French and his limited proficiency in Dutch, we had another colleague present to help translate.
Youssefs main frustration during work would be unexpected delays. These happen often, and Youssef gets frustrated losing his precious time. “I have a life, you know”. He welcomes the idea of pinning obstacles and alerting standstills.
Youssef has a good bond with his colleagues, although he senses a division between the languages. He thinks an app for this type of crew should be ‘without language’, focusing mainly on visual display.

Dimitri, 51, prefers being called by his nickname ‘Rizzo’:
truck driver. After losing his job as a
mailman, Rizzo decided to restrict alcohol use to his personal life. Working as a garbage truck driver was his big second chance. 11 years at Renewi; he loves his job. He knows ‘his’ route by heart and takes pride in that, and over the years has accumulated a lot of valuable knowledge about ‘the lay of the land’, for which he is valued by coworkers and managers alike. Dimitri likes the idea of an app, as long as it is easy to work with.
When I mentioned
the external controls he got more enthousiast. Dimitri likes the transferring route feature, because he likes offering and receiving help. Dimitri gets frustrated by communication, especially when there are no clear protocols in place or when ‘he has to do all the worlk’. To illustrate he told the following story: The garbage you put out should be of a maximum volume of 1m3. One time, a restaurant put out their cardboard waste: it spanned almost the entire sidewalk at 8meters long,and about 3 meters in height.

The current procedure for these situations is that the driver takes pictures with his personal phone, then sends these to administration. Dimitri thinks using a personal phone is unfair, as nobody gets compensated for that, plus unsafe, and tedious, and whatnot. On top of that, he has to call administration! The driver makes the end decision to either take or leave the waste, but administration needs to get notified with the location, picture, and decision, because in any case the garbage owner gets a fine for violating the rules. But the driver has to do all the extra work, sometimes it is unclear which house the garbage belongs to so he has to look for the owner, and remember to send the picture on time, and the trouble of making a phone call, and all this extra stuff, as if they don’t have enough work already! He suggests a reporting feature, that automatically sends the pic/pin to the administration,

References

-, D. (2023, January 28). *Intro to Mobile-Pak<sup>TM* by Soft-PakTM [Video]. Vimeo. <span>https://vimeo.com/456756495?embedded=true

-, T. (n.d.). *open-source-spec/osrm-vs-valhalla.md at master · Telenav/open-source-spec*. GitHub. https://github.com/Telenav/open-source-spec/blob/master/osrm/doc/osrm-vs-valhalla.md

🇺🇦 M. K. (2022, April 30). *Extracting Open Street Map (OSM) street data from data files using PyOsmium*. Medium. https://max-coding.medium.com/extracting-open-street-map-osm-street-data-from-data-files-using-pyosmium-afca6eaa5d00

17 Best Free Route Planners with Unlimited Stops. (2023, March 29). Badger Maps. https://www.badgermapping.com/blog/free-route-planner-with-unlimited-stops/#gmaps

About Renewi. (n.d.). https://indd.adobe.com/view/6b11f274-0267-40bf-8f7a-d7c0ed437d53

addaxmotors.com, A. (2023, March 1). *Elektrische afvalcontainer en vuilniswagen*. Addax Motors. https://www.addaxmotors.com/nl/configurations/afvalcontainer

Afvalbeleid. (n.d.). Interafval. https://interafval.be/afvalbeleid

Algorithms used by the ArcGIS Network Analyst extension—ArcMap | Documentation. (n.d.). https://desktop.arcgis.com/en/arcmap/latest/extensions/network-analyst/algorithms-used-by-network-analyst.htm

Asaduzzaman, M., Geok, T. K., Hossain, F., Sayeed, S., Abdaziz, A., Wong, H. Y., Tso, C. P., Ahmed, S., & Bari, A. (2021). An Efficient Shortest Path Algorithm: Multi-Destinations in an Indoor Environment. *Symmetry*, *13*(3), 421. https://doi.org/10.3390/sym13030421

Atuhaire, R. (2022, June 23). *Popular Shortest Path Algorithms*. Ronnie Atuhaire's Blog 🤓. https://blog.octachart.com/popular-shortest-path-algorithms

Blue Elephant. (2022, June 14). *History of garbage trucks. The 20 oldest garbage trucks from the past.* [Video]. YouTube. https://www.youtube.com/watch?v=0GQtpndR50I

Bmkmanoj. (n.d.). *geeksforgeeks/dynamic_programming.md at master · bmkmanoj/geeksforgeeks*. GitHub. https://github.com/bmkmanoj/geeksforgeeks/blob/master/makethebook/dynamic-programming/dynamic_programming.md

Boeckem, B. (2021, August 3). *Three Ways Autonomous Technology Can (And Will) Transform Your Business*. Forbes. https://www.forbes.com/sites/forbestechcouncil/2021/08/03/three-ways-autonomous-technology-can-and-will-transform-your-business/

Chandra, V. (2021, December 13). *Routing Optimization for Garbage Collection - Analytics Vidhya - Medium*. Medium. https://medium.com/analytics-vidhya/routing-optimization-for-garbage-collection-495225a67daf

Computer Science - Error 404 | California State University Stanislaus. (n.d.). https://www.cs.csustan.edu/%7Emmartin/teaching/CS4960S15/Corey+Trevena+-+Pathfinding+Algorithms+in+Navigational+Meshes+PDF.pdf

Cordeau, J., & Laporte, G. (2007, May 5). *The dial-a-ride problem: models and algorithms*. Annals of Operations Research; Springer Science+Business Media. https://doi.org/10.1007/s10479-007-0170-8

Corey Trevena - Pathfinding Algorithms in Navigational Meshes. (n.d.). *https://www.cs.csustan.edu/*. https://www.cs.csustan.edu/%7Emmartin/teaching/CS4960S15/Corey+Trevena+-+Pathfinding+Algorithms+in+Navigational+Meshes+PDF.pdf

Decimal degrees - GIS Wiki | The GIS Encyclopedia. (n.d.). http://wiki.gis.com/wiki/index.php/Decimal_degrees

Derecia, U., & Karabekmez, M. E. (2022, August 1). *The applications of multiple route optimization heuristics and meta-heuristic algorithms to solid waste transportation: A case study in Turkey*. Decision Analytics Journal; Elsevier BV. https://doi.org/10.1016/j.dajour.2022.100113

Dijkstra's algorithm. (2023, April 6). *Dijkstra's algorithm*. Dijkstra's Algorithm. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Din, H. T. E. D. (2021, June). *Comparative Analysis of Ant Colony Optimization and Genetic Algorithm in Solving the Traveling Salesman Problem*. diva-portal.org.

Directions API Playground | Developer Playgrounds. (n.d.). Mapbox. https://docs.mapbox.com/playground/directions/

DoverESG. (2023, April 11). *Soft-Pak Map-Pak Route Software* [Video]. Vimeo. https://vimeo.com/733663235?embedded=true&source=vimeo_logo&owner=36386144

Druktebarometer Info Gent. (2021a, March 12). https://data.stad.gent/explore/dataset/druktebarometer-info/api/

Druktebarometer Info Gent. (2021b, March 12). https://data.stad.gent/explore/dataset/druktebarometer-info/api/

E-truck Europe. (2017, December 14). *Hydrogen powered garbage truck from CURE present at ReinigingsDemoDagen - Etrucks Europe*. Etrucks Europe. https://e-truckseurope.com/en/hydrogen-powered-garbage-truck-from-cure-present-at-reinigingsdemodagen/

EU climate ambitions spell trouble for electricity from burning waste. (2021, May 26). Clean Energy Wire. https://www.cleanenergywire.org/news/eu-climate-ambitions-spell-trouble-electricity-burning-waste

Explore — Open Data Portaal - Stad Gent. (n.d.). https://data.stad.gent/explore/?disjunctive.keyword&disjunctive.theme&sort=modified

Falling Concrete and Sturdy Dykes. The Economic Differences Between Belgium and the Netherlands. (n.d.). The Low Countries. https://www.the-low-countries.com/article/falling-concrete-and-sturdy-dykes-the-economic-differences-between-belgium-and-the-netherlands

Figure 2 | Optimal Routing of Solid Waste Collection Trucks: A Review of Methods. (n.d.). https://www.hindawi.com/journals/je/2018/4586376/fig2/

folium. (2022, December 12). PyPI. https://pypi.org/project/folium/

Fragapane, R. (2022, April 28). *The 11 Best Free Route Planners with Unlimited Stops*. Maptive. https://www.maptive.com/11-best-free-route-planners-with-unlimited-stops/

Francois-Rozet. (n.d.). *GitHub - francois-rozet/mv-tsp: Stochastic improvement of deterministic algorithms for MV-TSP*. GitHub. https://github.com/francois-rozet/mv-tsp

GeeksforGeeks. (2022, September 29). *Genetic Algorithms*. https://www.geeksforgeeks.org/genetic-algorithms/

GeeksforGeeks - Brute Force. (2021, May 4). *Brute Force Approach and its pros and cons*. GeeksforGeeks. https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

Get adjacent ways Overpass API. (n.d.). Stack Overflow. https://stackoverflow.com/questions/75558305/get-adjacent-ways-overpass-api?rq=1

Ghadami, N. (2021, June 28). *Smartening the movement path of municipal garbage trucks using genetic algorithm with emphasis on economic-environmental indicators*. https://www.peertechzpublications.com/articles/AEST-5-141.php

Gibbard, J. (n.d.). *Using OpenStreetMap Offline*. https://www.gibbard.me/openstreetmap/

GitHub - osmlab/awesome-openstreetmap: 😎 Curated list of awesome OpenSteetMap-projects (By awesome-openstreetma [osmlab]). (n.d.). GitHub. https://github.com/osmlab/awesome-openstreetmap

GmbH, G. (2022, November 3). *3 Ways to get OpenStreetMap(OSM) Data*. Geoapify. https://www.geoapify.com/ways-to-get-openstreetmap-data/

Google OR-Tools versus OptaPlanner comparison. (n.d.). OptaPlanner. https://www.optaplanner.org/competitor/or-tools.html

GPS Leaflet Tracking Software for leaflet distribution. (n.d.). Trackit247. https://trackit247.com/software-solutions/gps-leaflet-tracking-software/

Grundhauser, E. (2016, March 17). *The Evolution of the Garbage Truck*. Atlas Obscura. https://www.atlasobscura.com/articles/the-evolution-of-the-garbage-truck

Heroku vs Netfliy vs Vercel vs GitHub Pages vs Firebase vs Vercel - Ritza Articles. (n.d.). https://ritza.co/articles/heroku-vs-netfliy-vs-vercel-vs-github-pages-vs-firebase-vs-vercel/

How do I develop an Android app on PC? (n.d.). Quora. https://www.quora.com/How-do-I-develop-an-Android-app-on-PC

How to make waste collection more efficient with Sygic. (n.d.). Sygic | Bringing Life to Maps. https://www.sygic.com/blog/2019/how-to-make-waste-collection-more-efficient-with-sygic

IBGE, I. (2011). *Brussels Waste Management Plan with waste reduction targets, Belgium*. https://www.acrplus.org/images/project/Pre-waste/Best_Practices/

Pre_waste_103_BE_Brussels_Waste_Plan_reduction_targets_23_11_2011.pdf

If a supercomputer can test 100,000 trillion combinations per second, how long would it take to test a vigintillion combinations? (n.d.). Quora. https://www.quora.com/If-a-supercomputer-can-test-100-000-trillion-combinations-per-second-how-long-would-it-take-to-test-a-vigintillion-combinations

Interline Technologies. (n.d.). *Valhalla multi-purpose routing engine - help, consulting, instant download of Tilepacks - Interline Technologies*. Interline. https://www.interline.io/valhalla/

Introduction | React Leaflet. (n.d.). https://react-leaflet.js.org/docs/start-introduction/

Ivla. (2023). *Ivla*. Ivla.be. https://www.ivla.be/swfiles/files/520320_IVLA_kalender-2023-RONSE.pdf

IVLA I Intergemeentelijke Vereniging IVLA. (n.d.). https://www.ivla.be/

Khamis, A. K. (2021). *Getting Started — AI Search Algorithms for Smart Mobility*. https://smartmobilityalgorithms.github.io/book/content/GettingStarted/index.html

Kowalik, Ł., Li, S., Nadara, W., Smulewicz, M., & Wahlström, M. (2022a, March 1). *Many-visits TSP revisited*. Journal of Computer and System Sciences; Elsevier BV. https://doi.org/10.1016/j.jcss.2021.09.007

Kowalik, Ł., Li, S., Nadara, W., Smulewicz, M., & Wahlström, M. (2022b, March 1). *Many-visits TSP revisited*. Journal of Computer and System Sciences; Elsevier BV. https://doi.org/10.1016/j.jcss.2021.09.007

Kumar, R. (2021, January 6). *List of Free and Open Source Maps and Global Positioning System (GPS)*. DevOpsSchool.com. https://www.devopsschool.com/blog/list-of-free-and-open-source-maps-and-global-positioning-system-gps/

Lathrop, R. L. (2020). *Local search algorithms*. https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/2020_SS1_CS171/chap_4_Local_Search.pdf

Layman Report 2021. (n.d.). *www.waterstofnet.eu*. https://www.waterstofnet.eu/_asset/_public/Lifeandgrabhy/Lifengrabhy_LaymanReport_2021_DEF.pdf

Liedman, P. L. (2015). *Leaflet Routing Machine*. https://tomickigrzegorz.github.io/leaflet-examples/#63.leaflet-routing-machine

Live Server - Visual Studio Marketplace. (n.d.). https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

Mapbox. (2018, May 30). *9 Years of OpenStreetMap GPS Tracks Available for Mapping*. Medium. https://blog.mapbox.com/9-years-of-openstreetmap-gps-tracks-available-for-mapping-47f6074c0688

Maskell, R. (2022, June 14). *Google Maps: How to Plan a Route with More than 10 Destinations*. WinBuzzer. https://winbuzzer.com/2021/02/12/how-to-plan-a-route-with-more-than-10-destinations-on-google-maps-xcxwbt/

MBTiles - OpenStreetMap Wiki. (n.d.). https://wiki.openstreetmap.org/wiki/MBTiles

McGill MMA - Garbage Collection Optimization Project - Montreal. (n.d.). [Video]. YouTube. https://youtube.com/watch?time_continue=142&v=nUR1DdhX8z0&embeds_widget_referrer=https%3A%2F%2Fmedium.com%2F&embeds_euri=https%3A%2F%2Fcdn.embedly.com%2F&embeds_origin=https%3A%2F%2Fcdn.embedly.com&source_ve_path=MTM5MTE3LDEyNzI5OSwxMzkxMTcsMTI3Mjk5&feature=emb_logo

Mitchell, J. S. B. (2000, January 1). *Geometric Shortest Paths and Network Optimization*. Elsevier eBooks; Elsevier BV. https://doi.org/10.1016/b978-044482537-7/50016-4

Naru, B. (n.d.). *GitHub - barbeau/awesome-gnss: Community list of open-source GNSS software and resources*. GitHub. https://github.com/barbeau/awesome-gnss

NEOS Guide. (2022, August 1). *Multiple Traveling Salesman Problem (mTSP) - NEOS Guide*. https://neos-guide.org/case-studies/tra/multiple-traveling-salesman-problem-mtsp/

OpenStreetMap Foundation. (n.d.). https://wiki.osmfoundation.org/wiki/Main_Page

OpenStreetMap Layers. (2019). *OpenStreetMap Layers*. https://www.arcgis.com/home/group.html?id=66d66956ab444ae89e8265f008704d4b#overview

OSM Lab. (n.d.). GitHub. https://github.com/osmlab

OSM QA tiles country extracts. (n.d.). https://osmlab.github.io/osm-qa-tiles/country.html

overpass. (2019, December 10). PyPI. https://pypi.org/project/overpass/

overpass turbo. (n.d.). https://overpass-turbo.eu/

Overpass turbo/Examples - OpenStreetMap Wiki. (n.d.). https://wiki.openstreetmap.org/wiki/Overpass_turbo/Examples

Page not found · GitHub Pages. (n.d.). https://smartmobilityalgorithms.github.io/book/content/ToolsAndPythonLibraries/index.htm

Pathfinding. (2023, March 23). *Pathfinding*. Wikipedia. https://en.wikipedia.org/wiki/Pathfinding

Python Overpass, M. (n.d.). *GitHub - mvexel/overpass-api-python-wrapper: Python bindings for the OpenStreetMap Overpass API*. GitHub. https://github.com/mvexel/overpass-api-python-wrapper

RapidAPI. (n.d.). *Feroeg - Reverse Geocoding API Documentation (castelli0giovanni-VdUSmLXuCR3)*. https://rapidapi.com/castelli0giovanni-VdUSmLXuCR3/api/feroeg-reverse-geocoding/

Reichel, A. R. (2013). *Municipal waste management in Belgium*. eea.europa.eu. https://www.eea.europa.eu/publications/managing-municipal-solid-waste/

Renewi: A leading waste-to-product company. (n.d.). Renewi. https://www.renewi.com/en/

RouteSmart. (2015, August 17). *Multi Day Routing for Commercial Waste Collection* [Video]. YouTube. https://www.youtube.com/watch?v=VDokIqewibY

Routing problems — Mathematical Optimization. (2019, January 18). https://scipbook.readthedocs.io/en/latest/routing.html#capacitated-vehicle-routing-problem

Saurabh. (2019, April 13). *WebSocket vs HTTP Calls - Performance Study: WebSocket vs HTTP Calls*. Browsee. https://browsee.io/blog/websocket-vs-http-calls-performance-study/

Sensoneo. (2022, December 22). *Driver Navigation App for Waste Vehicles | SENSONEO*. https://sensoneo.com/driver-navigation-waste-vehicles/

Shankar, S. (2020a, August 29). *Route Planning for Garbage collection with IoT – Part 1*. IoTEDU. https://iot4beginners.com/route-planning-for-garbage-collection-with-iot-part-1/

Shankar, S. (2020b, August 29). *Route Planning for Garbage collection with IoT – Part 1*. IoTEDU. https://iot4beginners.com/route-planning-for-garbage-collection-with-iot-part-1/

Sharma, K. S., & Munshi, C. M. (n.d.). *A Comprehensive and Comparative Study Of Maze-Solving Techniques by Implementing Graph Theory*.

Shortest Paths — NetworkX 3.1 documentation. (n.d.). https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html

Stavropoulos, P., Foteinopoulos, P., & Papapacharalampopoulos, A. (2021, August 23). *On the Impact of Additive Manufacturing Processes Complexity on Modelling*. Applied Sciences; MDPI. https://doi.org/10.3390/app11167743

Sulemana, A. (2018, October 9). *This paper reviews the effect of applying optimization methods on the collection process of solid waste, with particular interest in mathematical programming and geographic information system approaches in developing countries. Mathematical programming approaches maximize or minimize an objective function for improvement in procedure, to ensure operational efficiency and also serve as decision support tools. They however provide partial solutions when implemented in reality and cannot fully handle road network constraints. Geographic information system approaches allow processing of additional considerations, often ignored in other methods, such as the street network modeling. Incorporating environmental pollution consideration is very challenging in this approach, the vehicle routing problem solver encountering limits for large data. For enhanced efficiency of the vehicle routing systems, studies should further focus on incorporating all network constraints, environmental pollution considerations, and impact of land use changes on routing.* https://www.hindawi.com/journals/je/2018/4586376/

Team, I. (n.d.). *GitHub - ionic-team/ionic-framework: A powerful cross-platform UI toolkit for building native-quality iOS, Android, and Progressive Web Apps with HTML, CSS, and JavaScript.* GitHub. https://github.com/ionic-team/ionic-framework

Telematics. (2023, April 1). *Telematics*. Wikipedia. https://en.wikipedia.org/wiki/Telematics

Terravision. (n.d.). https://artcom.de/en/?project=terravision

the Guardian. (2020, February 3). Five reasons Belgium has the worst traffic in Europe. *The Guardian*. https://www.theguardian.com/cities/2014/aug/28/belgium-worst-traffic-europe-brussels-antwerp-congestion

The Ultimate Guide to Smart Waste Management. (2023, February 13). Greener and Smarter Waste - Nordsense. https://nordsense.com/the-ultimate-guide-to-smart-waste-management/#:~:text=What%20is%20Smart%20Waste%20Management,waste%20generation%20patterns%20and%20behavior.

Tools and Python Libraries — AI Search Algorithms for Smart Mobility. (n.d.). https://smartmobilityalgorithms.github.io/book/content/ToolsAndPythonLibraries/index.html

Travelling salesman problem. (2023, March 23). *Travelling salesman problem*. Wikipedia. https://en.wikipedia.org/wiki/Travelling_salesman_problem

Valhalla Docs. (n.d.). https://valhalla.github.io/valhalla/

Valhalla FOSSGIS Server Demo App. (n.d.). https://valhalla.openstreetmap.de/

Walsh, D. (2014, February 12). *WebSocket and Socket.IO*. David Walsh Blog. https://davidwalsh.name/websocket

What is the Difference Between GNSS and GPS? - everything RF. (n.d.). https://www.everythingrf.com/community/what-is-the-difference-between-gnss-and-gps_58

Wikipedia - Travelling salesman problem. (2023, March 23). *Travelling salesman problem*. Wikipedia. https://en.wikipedia.org/wiki/Travelling_salesman_problem

Wikipedia contributors. (2023a, January 24). *Geo-fence*. Wikipedia. https://en.wikipedia.org/wiki/Geo-fence

Wikipedia contributors. (2023b, March 23). *Pathfinding*. Wikipedia. https://en.wikipedia.org/wiki/Pathfinding

Will “Garbage Truck Driver” Be Automated or Replaced By Robots? (n.d.). https://www.replacedbyrobot.info/59434/garbage-truck-driver

Writer, S. O. A. (2019, September 12). *Reducing the pollution from garbage trucks*. Athens Banner-Herald. https://eu.onlineathens.com/story/opinion/letters/2019/09/11/letter-reducing-pollution-from-garbage-trucks-would-improve-athenians-health/3051977007/

Zoom levels - OpenStreetMap Wiki. (n.d.). https://wiki.openstreetmap.org/wiki/Zoom_levels