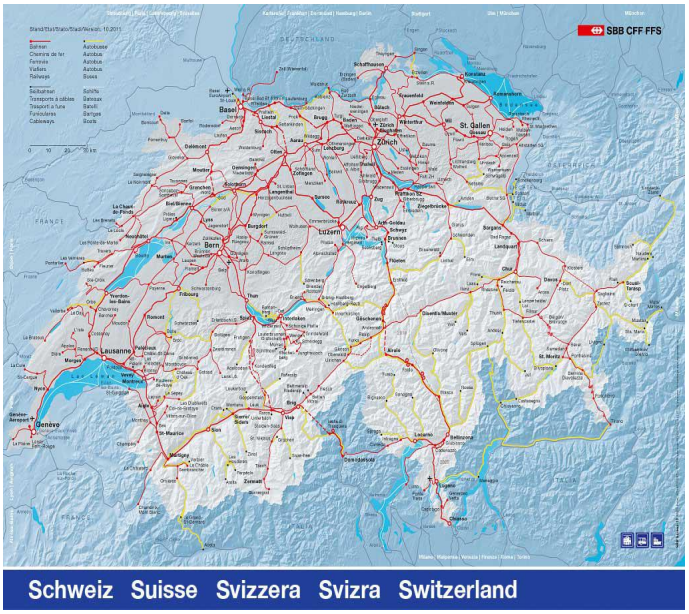
h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Synchronisation des Arrêts / Points d'Arrêt entre OSM et ATLAS



Travail de bachelor présenté par

Guillem MASSAGUÉ Querol

Informatique et systèmes de communication avec orientation
Sécurité Informatique

Juin, 2025

Professeur-e HES responsable

Orestis MALASPINAS

Mandant

Matthias GÜNTER

Carte du réseau ferroviaire suisse.

Source : CFF.

TABLE DES MATIÈRES

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

RÉSUMÉ

Ce projet a pour objectif de synchroniser les arrêts de transport public présents dans OpenStreetMap avec ceux du système officiel suisse ATLAS, afin d'améliorer la précision et la fiabilité des données. Dans un premier temps, nous analysons la structure, la couverture et les balises associées des deux jeux de données, ATLAS et OpenStreetMap, spécifiquement en Suisse.

Le cœur de notre démarche repose sur un processus de correspondance sophistiqué combinant plusieurs méthodes : correspondance exacte, par nom, par distance, par itinéraire, etc. Cette approche permet d'identifier avec précision les arrêts communs aux deux bases de données. Une première analyse statistique des résultats obtenus est ensuite réalisée.

Face aux nombreux cas problématiques rencontrés, nous avons développé une application web conviviale permettant de visualiser simultanément les deux ensembles de données ainsi que leurs correspondances. Cet outil facilite l'identification des incohérences, la génération de rapports détaillés et l'exécution d'ajustements manuels pour corriger les divergences.

Le projet se conclura par la présentation des résultats finaux, définissant une stratégie claire pour atteindre une synchronisation complète entre OpenStreetMap et ATLAS. Cela garantira un ensemble de données de transport public cohérent, fiable et exploitable facilement pour divers usages.

Candidat-e :

GUILLEM MASSAGUÉ QUEROL

Filière d'études : ISC

Professeur-e(s) responsable(s) :

ORESTIS MALASPINAS

En collaboration avec : SKI+

Travail de bachelor soumis à une convention de stage en entreprise : non

Travail soumis à un contrat de confidentialité : non

ACRONYMS

GTFS General Transit Feed Specification

OSM OpenStreetMap

UIC Union Internationale des Chemins de Fer

LISTE DES ILLUSTRATIONS

LISTE DES TABLEAUX

INTRODUCTION

CONTEXTE ET PROBLÉMATIQUE

La digitalisation des services de mobilité a rendu la qualité et la fiabilité des données de transport public plus cruciales que jamais. En Suisse, les systèmes d'information aux voyageurs, les outils de planification de réseau et les applications de navigation reposent sur des référentiels géographiques précis des arrêts et points d'arrêt. Le système de référence officiel pour ces données est la base de données ATLAS, qui fait autorité en la matière. Parallèlement, OpenStreetMap (OSM), un projet cartographique mondial et collaboratif, s'est imposé comme une source de données extrêmement riche et fréquemment utilisée pour la représentation cartographique et diverses applications tierces, grâce à sa flexibilité et sa couverture exhaustive.

Le défi majeur, et le cœur de ce travail de Bachelor, réside dans la divergence systémique entre ces deux jeux de données. Bien qu'ils décrivent la même réalité physique — le réseau de transport public —, ils le font avec des coordonnées, des identifiants et des hiérarchies qui ne sont pas nativement synchronisés. Ces écarts, qu'ils soient de quelques mètres ou plus significatifs, engendrent des incohérences problématiques :

— **Pour les usagers :** informations contradictoires, localisation erronée des arrêts sur les applications, et une expérience de voyage dégradée.
— **Pour les exploitants et planificateurs :** difficultés dans la planification des lignes, optimisation des correspondances et gestion de l'infrastructure.

Ce projet de Bachelor s'attaque directement à cette problématique. Il vise à concevoir et mettre en œuvre une approche systématique pour identifier, analyser et corriger les discordances entre les données d'arrêts d'ATLAS et d'OSM en Suisse. L'ambition est de transformer deux sources de données parallèles en un écosystème informationnel cohérent et fiable.

Sauf indication contraire, toutes les statistiques et cartes de ce mémoire (y compris l'introduction) sont calculées sur le snapshot de données du 11 août 2025.

Objectifs

Les objectifs principaux de ce travail sont les suivants :

— Concevoir et valider une méthodologie robuste pour la comparaison et la correspondance

automatisée des données d'arrêts entre ATLAS et OSM.

— Identifier et quantifier les différents types d'incohérences (écarts de position, absences de correspondance, etc.).

— Développer un outil d'aide à la décision et à la correction pour traiter les cas ambigus qu'un algorithme ne peut résoudre seul.

— Mettre en œuvre les corrections nécessaires et générer des rapports consolidés pour documenter les interventions.

— Contribuer à l'amélioration durable de la qualité des données de transport public, au bénéfice de l'ensemble des acteurs de la mobilité.

APPROCHE MÉTHODOLOGIQUE

Pour atteindre ces objectifs, notre démarche s'articule en deux phases complémentaires, alliant traitement automatisé et validation humaine supervisée.

Phase 1 : Traitement automatisé

La première phase repose sur l'utilisation de scripts en langage Python pour automatiser les tâches de traitement, de nettoyage et de comparaison des deux jeux de données. Ces scripts mettent en œuvre différents algorithmes de correspondance, allant de la simple proximité géographique à des méthodes plus sophistiquées combinant plusieurs attributs (comme le nom de l'arrêt) pour établir des paires de correspondance potentielles entre les entités ATLAS et OSM.

Phase 2 : Application web interactive

Cependant, la complexité des données et la présence de cas ambigus ont rapidement mis en évidence les limites d'une approche 100 % automatique. Pour surmonter cet obstacle, la seconde phase a consisté à développer une application web interactive. Dotée d'un backend en Python (framework Flask), d'une base de données MySQL pour la persistance des données et d'une interface en JavaScript, cette application remplit un double rôle :

Visualisation : Elle offre une représentation cartographique claire des correspondances trouvées, des divergences et des entités non appariées.

Validation et Correction : Elle fournit une interface de gestion permettant à un opérateur humain d'examiner les cas problématiques, de valider les suggestions de l'algorithme

et d'appliquer manuellement des solutions correctives adaptées à chaque type d'incohérence.

Code source

Ce document ne présentera que des extraits de code ciblés pour illustrer des points spécifiques. L'intégralité du code source développé dans le cadre de ce projet est disponible sur le dépôt Git suivant :

```
https://githepia.hesge.ch/guillem.massague/bachelor-project
```

Conventions de couleur

Note sur les figures : Dans les captures d'écran de l'application web présentées dans ce mémoire, les conventions de couleur sont les suivantes :

- **Points verts** : arrêts ATLAS avec une correspondance OSM confirmée
- **Points bleus** : nœuds OSM correspondants
- **Points rouges** : arrêts ATLAS sans correspondance
- **Points gris** : nœuds OSM sans correspondance

Structure du mémoire

Ce mémoire est structuré de manière à suivre la progression logique de notre recherche, depuis l'analyse des données brutes jusqu'à la validation des résultats finaux.

Chapitres 1 et 2 : Présentation des jeux de données. Nous commencerons par une analyse détaillée des sources de données ATLAS et OSM. Nous décrirons leur structure, leurs attributs, leurs forces et leurs limitations respectives, qui constituent le fondement de notre problématique.

Chapitre 3 : Premières approches de correspondance. Ce chapitre explorera les méthodes initiales et les plus directes pour apparier les arrêts, principalement basées sur la proximité géographique. Nous y évaluerons les performances et les lacunes de ces techniques simples.

Chapitre 4 : Développement d'un algorithme de correspondance avancé. Forts des enseignements du chapitre précédent, nous décrirons ici la conception d'un algorithme plus

robuste, combinant plusieurs critères (distance, similarité textuelle, opérateur, itinéraire, etc.) pour améliorer la précision des correspondances automatiques.

Chapitre 5 : Analyse des écarts et des cas problématiques. Une fois les correspondances établies, ce chapitre se consacrera à l'analyse quantitative et qualitative des résultats. Nous y présenterons les statistiques sur les distances d'écart et nous catégoriserons les principaux types de problèmes rencontrés.

Chapitre 6 : Application web d'aide à la validation et à la correction. Nous présenterons l'outil web développé sur mesure. Ce chapitre détaillera son architecture technique, ses fonctionnalités de visualisation interactive, ainsi que l'interface de gestion conçue pour permettre une intervention humaine efficace et guidée.

Chapitre 7 : Analyse des résultats et validation. Ce chapitre évaluera l'efficacité de notre méthode combinée (algorithme et validation manuelle). Nous y quantifierons le nombre de corrections effectuées, l'amélioration de la qualité des données et la pertinence des solutions apportées.

Conclusion. Pour conclure, nous dresserons un bilan complet du projet, en synthétisant les apports et les résultats obtenus. Nous discuterons également des enseignements tirés, des limites de notre approche et des perspectives d'avenir pour l'amélioration continue des données de transport en Suisse.

CHAPITRE 1 : DONNÉES ATLAS

Les données ATLAS, au cœur de cette étude, proviennent de la plateforme Open Transport Data Swiss [1]. Cette ressource centralise les données des transports publics en Suisse, offrant une base précieuse pour l'analyse et le développement d'applications.
Sauf indication contraire, toutes les statistiques et cartes de ce chapitre sont calculées sur le snapshot de données du 11 août 2025.

Reproductibilité. Sauf mention contraire, toutes les figures et statistiques de ce chapitre ont été produites par des scripts reproductibles disponibles sous `memoire/scripts_used/`. Nous n'indiquons plus les scripts individuellement dans les légendes afin d'alléger la lecture.

1.1. ARRÊTS

La section sur les arrêts s'appuie sur le jeu de données traffic-points-actual-date [2]. Ce jeu de données recense les arrêts de transport public en Suisse, avec des informations détaillées sur leur localisation et leurs caractéristiques. Ces points peuvent être visualisés sur une carte interactive via l'application web https ://atlas.app.sbb.ch/ [3], offrant une représentation graphique intuitive des arrêts.

Nous nous concentrons ici sur deux colonnes principales : le numéro et la désignation des arrêts.

Le numéro d'un arrêt correspond à la référence UIC (Union Internationale des Chemins de fer), un standard international permettant d'identifier les lieux de transport public. Les deux premiers chiffres représentent le code du pays ; la Suisse, par exemple, utilise le code 85[4]. Ainsi, un numéro UIC comme "8502034" désigne un arrêt spécifique du réseau suisse.

La colonne "désignation" fait référence à une identification locale : une valeur de "3" peut, par exemple, indiquer que l'arrêt correspond à la plateforme 3 d'une gare.

Enfin, les données incluent également des informations sur l'opérateur responsable de chaque arrêt, un élément potentiellement utile pour établir des correspondances avec d'autres jeux de données.

Le jeu de données distingue deux types de `trafficPointElementType` : `BOARDING_AREA` et `BOARDING_PLATFORM`. Notre analyse se limite aux `BOARDING_PLATFORM`, car les `BOARDING_AREA` ne disposent pas de coordonnées géographiques. Pour extraire ces informations, nous

avons développé un script Python, `get_atlas_data.py`. Extrait simplifié du chargement/filtrage :

Extrait — `get_atlas_stops`

```python
def get_atlas_stops(output_path, download_url):
    response = requests.get(download_url)
    response.raise_for_status()
    with zipfile.ZipFile(io.BytesIO(response.content)) as z:
        csv_filename = z.namelist()[0]
        with z.open(csv_filename) as f:
            df = pd.read_csv(f, sep=';')
            # Suisse (UIC pays = 85) et coordonnees valides (WGS84)
            df = df[df['uicCountryCode'] == 85]
            df = df.dropna(subset=['wgs84North', 'wgs84East'])
            # Comptage des quais
            boarding_platforms = df[df['trafficPointElementType'] == '
    BOARDING_PLATFORM']
            df.to_csv(output_path, sep=';', index=False)
```

Statistiques ATLAS (WGS84). Sur l'instantané analysé :

- **Lignes avec coordonnées** : *56 510*.
- **`BOARDING_PLATFORM`** : *55 818*.
- **UIC distincts (`number`)** : *27 225*.
- **`designation` non vides** : *11 462* (*109* valeurs distinctes).
- **`designation` manquantes** : *44 356*, dont *4 499* cas où l'unique entrée du `number` est sans désignation.
- **Entrées identifiables par (`number`, `designation`) seul** : *10 928*.
- **Total identifiables par `number` + (`designation` ou unicité du `number`)** : *15 427*.

Remarque. Les valeurs manquantes de `designation` (très nombreuses) sont exclues du top pour éviter un effet disproportionné. Comme indiqué plus haut, 10 928 entrées sont déjà identifiables par le couple (`number`, `designation`) et 4 499 supplémentaires sont uniques par `number` seul.

Ce script filtre les données pour ne conserver que les entrées `BOARDING_PLATFORM` avec des coordonnées valides.

Concernant les coordonnées, le fichier fournit deux systèmes : le système de référence suisse

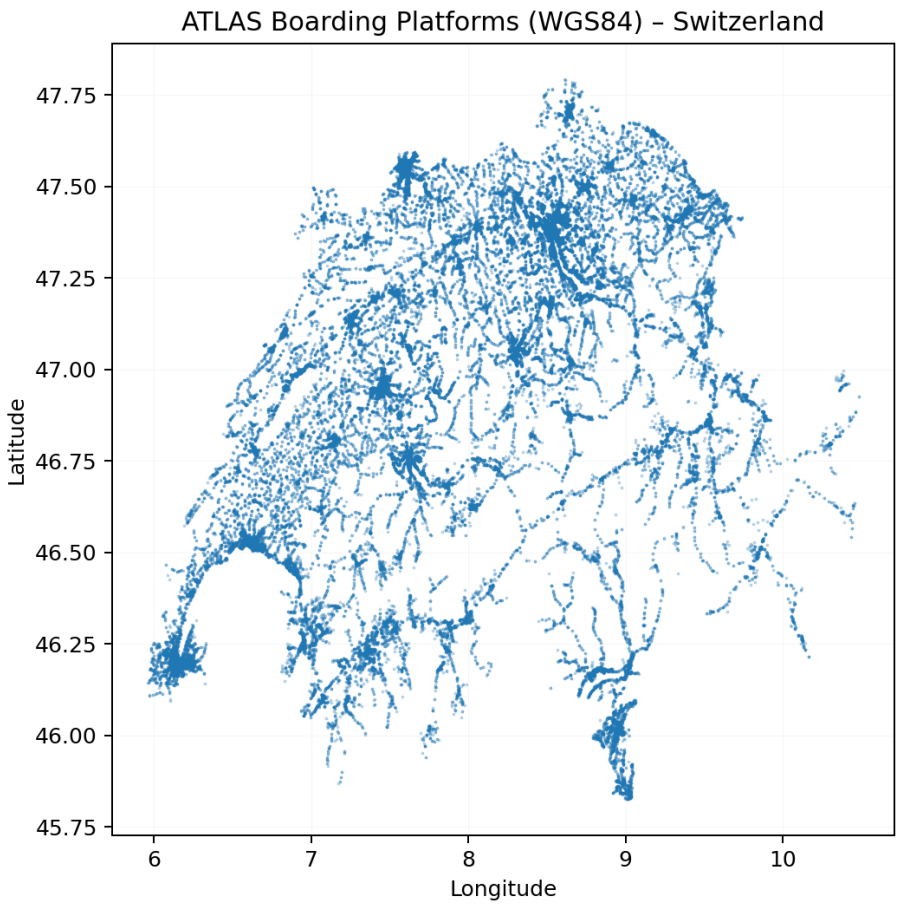


ILLUSTRATION 1.1 – ATLAS `BOARDING_PLATFORM` : distribution nationale (WGS84).

LV95 et le système de référence global WGS84. Étant donné que les données d'OpenStreetMap (OSM) utilisent les coordonnées WGS84, nous nous concentrerons uniquement sur cet ensemble de coordonnées pour le moment.

1.2. GTFS

Le General Transit Feed Specification (GTFS) est un format d'échange numérique initié par Google pour standardiser les horaires des transports publics et leurs informations géographiques, telles que la localisation des arrêts. En Suisse, ces données sont fournies par les entreprises de transport et publiées sur la plateforme OpenTransportDataSuisse. Elles servent à développer des applications pratiques, comme les outils de consultation d'horaires ou de planification de trajets.

Bien que notre projet se focalise actuellement sur la synchronisation des arrêts, les données GTFS relatives aux trajets suscitent également notre intérêt. Elles pourraient faciliter la correspondance entre les arrêts ATLAS et ceux d'OpenStreetMap, en exploitant les informations sur

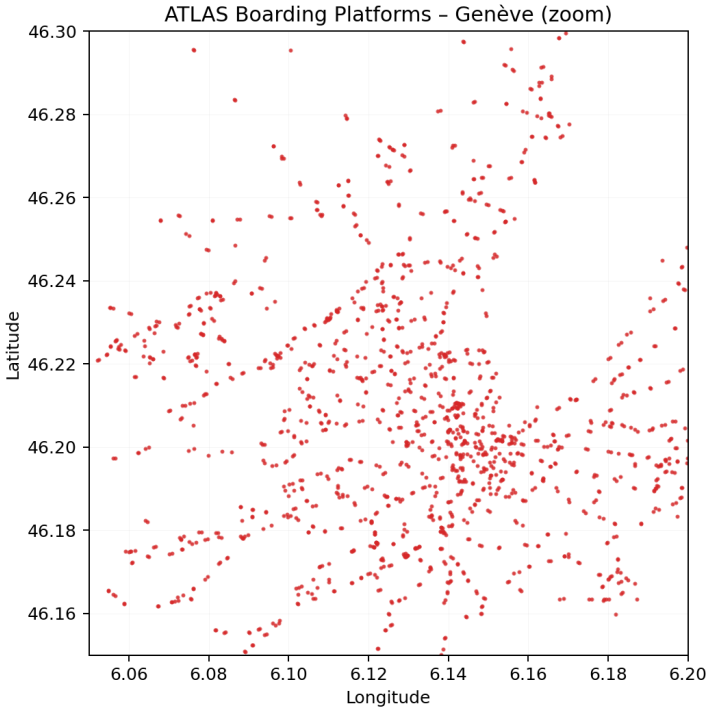


ILLUSTRATION 1.2 – ATLAS `BOARDING_PLATFORM` : zoom sur Genève.

les itinéraires présentes dans les deux ensembles de données. Parmi les nombreux fichiers GTFS disponibles, quatre retiennent notre attention : `stops.txt`, `stop_times.txt`, `routes.txt` et `trips.txt`.

a. Description des fichiers

Les fichiers GTFS suivants sont cruciaux pour notre analyse :

— **`stops.txt`** : Ce fichier répertorie les arrêts avec leurs coordonnées géographiques et d'autres attributs. Un extrait est présenté dans le tableau 1.1.

— **`routes.txt`** : Il décrit les lignes de transport, avec des informations comme le nom court, le nom long, et le type de transport. Voir le tableau 1.2.

— **`trips.txt`** : Ce fichier associe les trajets aux lignes et aux services. Un exemple est donné dans le tableau 1.3.

— **`stop_times.txt`** : Il contient les horaires d'arrivée et de départ pour chaque arrêt d'un trajet. Voir le tableau 1.4.

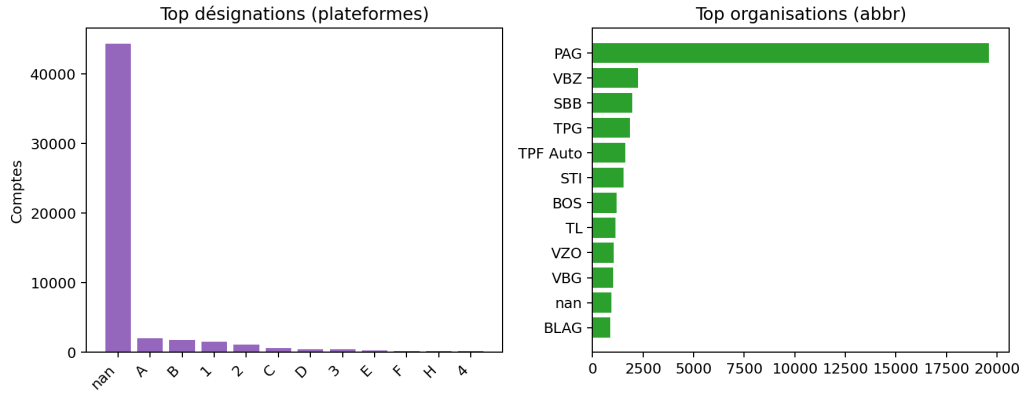


ILLUSTRATION 1.3 – À gauche : désignations de plateforme les plus fréquentes (hors valeurs manquantes). À droite : principales organisations (abrégés) déclarées.

TABLEAU 1.1 – Extrait du fichier `stops.txt`

stop_id	stop_name	stop_lat	stop_lon	parent_station
1101064	Malpensa Aeroporto, terminal 1	45.6272	8.7111	
8000339	Weissenhorn Eschach	48.3010	10.1351	
8000709 :0 :2	Neckarsulm Mitte	49.1935	9.2229	
8000778	Asselheim (D)	49.5762	8.1616	
8000781	Grünstadt-Nord	49.5734	8.1708	
8000988	Witzighausen	48.3174	10.0978	
8002015	Nördlingen	48.8508	10.4979	8002015P
8002015 :0 :4	Nördlingen	48.8509	10.4979	8002015P

b. Analyse et défis de jointure

Dans le script `get_atlas_data.py`,nous avons crée un fichier `routes_atlas.csv` qui associe les arrêts aux routes et directions qu'ils desservent à partir des données GTFS. On commence pour joindre les fichiers GTFS. Cette jointure s'appuie sur le fichier `stop_times.txt` pour connecter les arrêts aux trajets via `trip_id`, puis les trajets aux routes via `route_id`, en extrayant des paires uniques de route-direction par arrêt, accompagnées des noms des routes.

La mise en correspondance entre la colonne `stop_id` du fichier `stops.txt` (GTFS) et l'identifiant `sloid` d'ATLAS présente des défis significatifs. Premièrement, il n'existe pas de lien direct entre ces deux identifiants. Deuxièmement, les coordonnées géographiques des arrêts diffèrent entre les deux ensembles de données.

Exemple 1 : "Lancy-Pont-Rouge" :

Considérons la gare "Lancy-Pont-Rouge", opérée par les CFF. Dans le fichier `stops.txt` de

TABLEAU 1.2 – Extrait du fichier `routes.txt`

route_id	agency_id	route_short_name	route_desc	route_type
91-10-A-j22-1	37	10	T	900
91-10-B-j22-1	78	S10	S	109
91-10-C-j22-1	11	S10	S	109
91-10-E-j22-1	65	S10	S	109
91-10-F-j22-1	11	RE10	RE	106
91-10-G-j22-1	11	SN10	SN	109
91-10-j22-1	3849	10	T	900
91-10-Y-j22-1	82	IR	IR	103

TABLEAU 1.3 – Extrait du fichier `trips.txt`

route_id	trip_id	trip_short_name	direction_id
91-8-H-j25-1	994.TA.91-8-H-j25-1.59.R	6278	1
91-8-H-j25-1	995.TA.91-8-H-j25-1.59.R	2978	1
91-8-H-j25-1	996.TA.91-8-H-j25-1.59.R	2787	1
91-8-H-j25-1	997.TA.91-8-H-j25-1.59.R	4879	1
91-8-H-j25-1	998.TA.91-8-H-j25-1.59.R	10407	1
91-8-H-j25-1	999.TA.91-8-H-j25-1.59.R		1

GTFS, les données sont les suivantes :

Dans le fichier `traffic-points-actual-data`, on trouve :

Ici, le format de `stop_id` dans GTFS est `uic_number:0:local_ref`, où `uic_number` correspond à la colonne `number` dans ATLAS (8516155), et `local_ref` à `designation` (1 ou 2). Cela permet une correspondance, bien que les coordonnées géographiques divergent légèrement.

Exemple 2 : "Lausanne Bourdonnette" :

Prenons un deuxième exemple avec "Lausanne Bourdonnette". Dans `stops.txt` :

Et dans `traffic-points-actual-data` :

Dans ce cas, les désignations dans GTFS incluent "A", "B", "C", "D", ainsi que des références numériques comme "10000" et "10001", mais dans ATLAS, "A" n'a pas d'équivalent direct, et les références numériques ne sont pas assignées (lignes avec `designation` vide). Les coordonnées géographiques diffèrent également.

TABLEAU 1.4 – Extrait du fichier `stop_times.txt`

trip_id	arrival_time	departure_time	stop_id	stop_sequence
1.TA.1-9-j17-1.1.H	05 :25 :00	05 :25 :00	8502034 :0 :2	1
1.TA.1-9-j17-1.1.H	05 :28 :00	05 :29 :00	8502033 :0 :2	2
1.TA.1-9-j17-1.1.H	05 :33 :00	05 :33 :00	8502032 :0 :1	3
1.TA.1-9-j17-1.1.H	05 :36 :00	05 :36 :00	8502031 :0 :1	4
1.TA.1-9-j17-1.1.H	05 :42 :00	05 :42 :00	8502030 :0 :2	5
1.TA.1-9-j17-1.1.H	05 :50 :00	05 :50 :00	8502119 :0 :7	6
2.TA.1-9-j17-1.2.H	05 :53 :00	05 :53 :00	8502034 :0 :1	1
2.TA.1-9-j17-1.2.H	05 :57 :00	05 :58 :00	8502033 :0 :2	2

TABLEAU 1.5 – Extrait de `stops.txt` pour "Lancy-Pont-Rouge"

stop_id	stop_name	stop_lat	stop_lon	parent_station
8516155 :0 :1	Lancy-Pont-Rouge	46.18596197	6.12483039	Parent8516155
8516155 :0 :2	Lancy-Pont-Rouge	46.18595575	6.12495615	Parent8516155

c. Résulats

Nous générons un fichier intégré `atlas_routes_gtfs.csv` listant, par `sloid`, les couples `(route_id, direction_id)`. Sur notre jeu :

— **SLOIDs couverts par GTFS** : *32 248–32 249*.

— **Médiane des lignes par SLOID (GTFS)** : *2*.

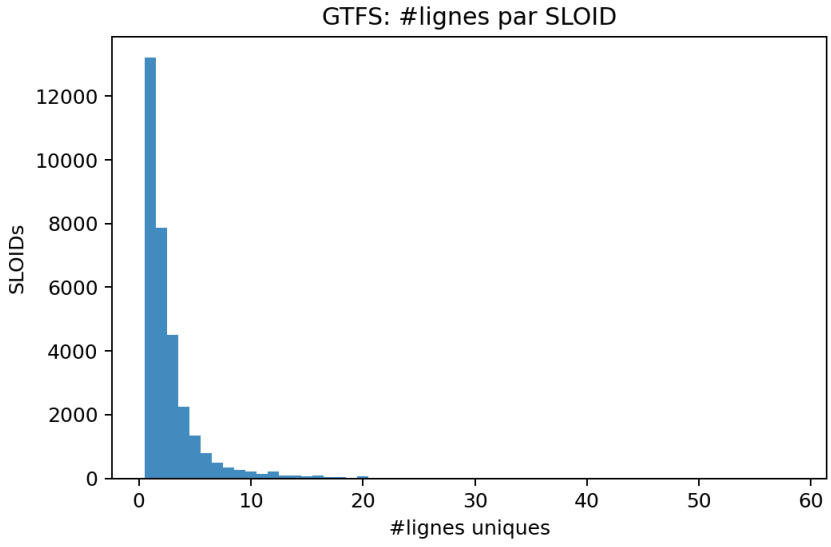


ILLUSTRATION 1.4 – GTFS : distribution du nombre de lignes par SLOID.

Motivation : ces couples route/direction sont une brique essentielle de notre *route matching*

TABLEAU 1.6 – Extrait de `traffic-points-actual-data` pour "Lancy-Pont-Rouge"

sloid	number	des.	wgs84East	wgs84North	designationOfficial
... :16155 :1 :1	8516155	1	6.12483137	46.18596333	Lancy-Pont-Rouge
... :16155 :1 :2	8516155	2	6.12495213	46.18595284	Lancy-Pont-Rouge

TABLEAU 1.7 – Extrait de `stops.txt` pour "Lausanne Bourdonnette"

stop_id	stop_name	stop_lat	stop_lon
8501210 :0 :10000	Lausanne, Bourdonnette	46.52342565	6.59074161
8501210 :0 :10001	Lausanne, Bourdonnette	46.52329585	6.58987025
8501210 :0 :A	Lausanne, Bourdonnette	46.52326494	6.58980736
8501210 :0 :B	Lausanne, Bourdonnette	46.52318459	6.58978940
8501210 :0 :C	Lausanne, Bourdonnette	46.52272720	6.58913363
8501210 :0 :D	Lausanne, Bourdonnette	46.52338238	6.59138840

OSM (détaillé au Chapitre 4).

d. Compléments cartographiques

Distances ATLAS–GTFS : définition et interprétation. Pour chaque `UIC` présent dans les deux jeux de données, nous calculons (i) le centroïde des arrêts GTFS associés et (ii) le centroïde des plateformes ATLAS correspondantes, puis la distance entre ces deux centroïdes. La médiane empirique de cette distance est $\sim 11{,}9\,m$ (53 912 UIC considérés), ce qui atteste d'une compatibilité géométrique globale satisfaisante. Les valeurs extrêmes reflètent principalement des différences de modélisation (positionnement des plateformes vs. points d'arrêt agrégés) et des cas de gares étendues.

Lecture du graphique. La figure ci-dessus présente l'histogramme de ces distances. Les petites distances dominent (alignement global), tandis que la queue lourde correspond à des gares complexes et à des divergences de granularité. Pour surmonter ces défis, nous avons exploité la structure de :

`stop_id` dans GTFS (`uic_number:0:local_ref`).

Nous utilisons les colonnes `number` et `designation` d'ATLAS pour établir une correspondance avec `uic_number` et `local_ref`, respectivement. La référence locale (`local_ref`) est normalisée (ex. "10000" devient "1", "10001" devient "2") pour obtenir `normalized_local_ref`. Les règles de correspondance sont les suivantes :

TABLEAU 1.8 – Extrait de `traffic-points-actual-data` pour "Lausanne Bourdonnette"

sloid	number	des.	wgs84East	wgs84North	designationOfficial
... :1210 :0 :1600	8501210		6.59074107	46.52342597	Lausanne, Bourdonnette
... :1210 :0 :1610	8501210		6.58986994	46.52329351	Lausanne, Bourdonnette
... :1210 :0 :1616	8501210	B	6.58979344	46.52318499	Lausanne, Bourdonnette
... :1210 :0 :2597	8501210	D	6.59138793	46.52338108	Lausanne, Bourdonnette
... :1210 :0 :2542	8501210	C	6.58913042	46.52272550	Lausanne, Bourdonnette

Associer si `uic_number` (GTFS) = `number` (ATLAS) et `normalized_local_ref` = `designation`. Si plusieurs entrées ATLAS existent pour un même `uic_number` : Associer à la seule entrée si elle est unique. Sinon, vérifier si `normalized_local_ref` = `designation` ou `normalized_local_ref` = dernière partie de `sloid` (ex. "ATLAS :8500010 :1" → "1"). Par exemple :

Pour "Lancy-Pont-Rouge", `stop_id` "8516155 :0 :1" donne `uic_number` "8516155" et `normalized_local_ref` "1", correspondant à `number` "8516155" et `designation` "1". Pour "Lausanne Bourdonnette", `stop_id` "8501210 :0 :B" donne `uic_number` "8501210" et `normalized_local_ref` "B", correspondant à `number` "8501210" et `designation` "B". Pour un `stop_id` "8500030 :0 :2" avec plusieurs entrées ATLAS, si `sloid` "ATLAS :8500030 :2" existe, il est associé car "2" correspond à `normalized_local_ref`.

Voici deux exemples concrets de correspondance :

GTFS : `stop_id="8505113:0:4"`, nom "Altdorf UR", `uic_number="8505113"`, `normalized_local_ref="4"` ; ATLAS : `sloid="ch:1:sloid:5113:2:4"`, nom "Altdorf UR", `number="8505113"`, `designation="4"` (match via `designation`). GTFS : `stop_id="8592669:0:C"`, nom "Carouge GE, Armes", `uic_number="8592669"`, `normalized_local_ref="C"` ; ATLAS : `sloid="ch:1:sloid:92669:0:241732"`, nom "Carouge GE, Armes", `number="8592669"`, `designation="C"` (match via `designation`, malgré `sloid` différent).

1.3. HRDF

Le *HAFAS Raw Data Format* structure des horaires exhaustifs. Deux fichiers sont clés : `BHFART` (SLOIDs gare/quai) et `GLEISE_WGS/LV95` (infrastructure et coordonnées de quai).

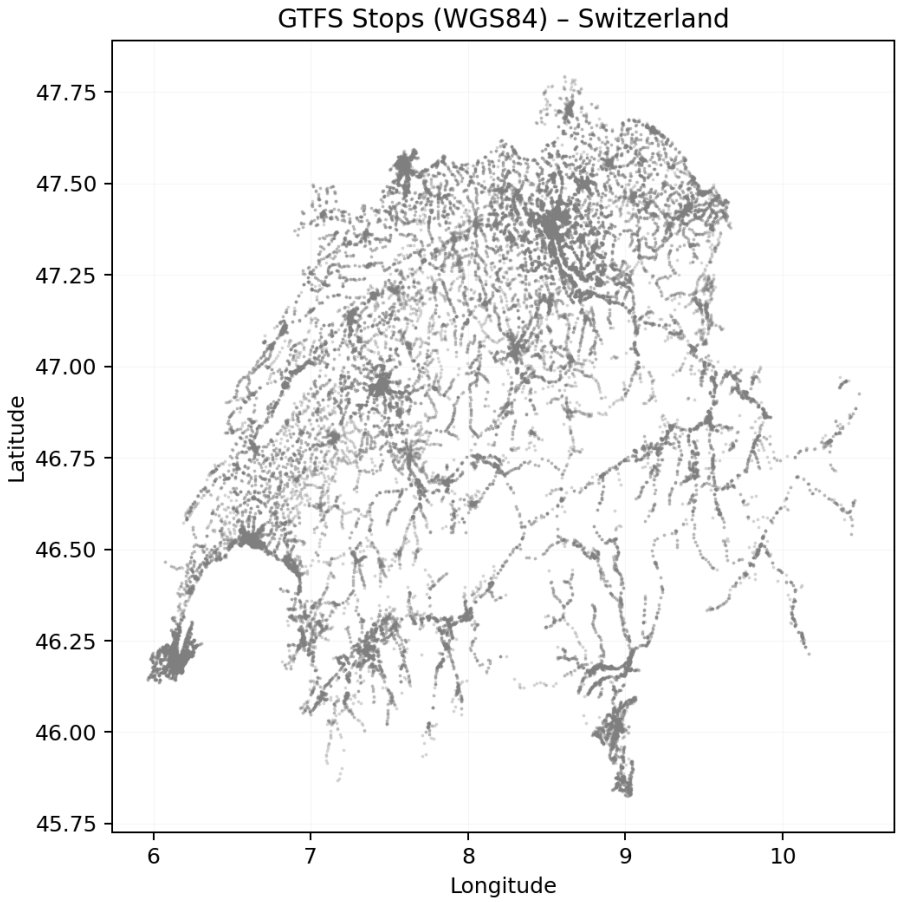


ILLUSTRATION 1.5 – Arrêts GTFS (Suisse) : 47 829 arrêts suisses détectés.

a. Comment nous le lisons

Notre extraction s'effectue en deux passes efficaces et ciblées :

1. **GLEISE_LV95 → paires (UIC, #ref) par `sloid`.** Nous parcourons `GLEISE_LV95` pour associer chaque `sloid` de quai au `UIC` de gare et au numéro de référence (#ref) qui identifie le quai.
2. **FPLAN → directions.** Pour ces (UIC, #ref) cibles seulement, nous analysons `FPLAN` afin d'extraire, par voyage, le premier et le dernier arrêt. En reliant ces arrêts à leurs noms (via `BAHNHOF`), nous formons des chaînes directionnelles *noms* (« Genève → Lausanne ») et *UIC* (« 8501008 → 8501120 »).

Ces chaînes seront utilisées au Chapitre 4 pour améliorer le *route matching* en absence d'identifiants de direction GTFS directement disponibles sur OSM.

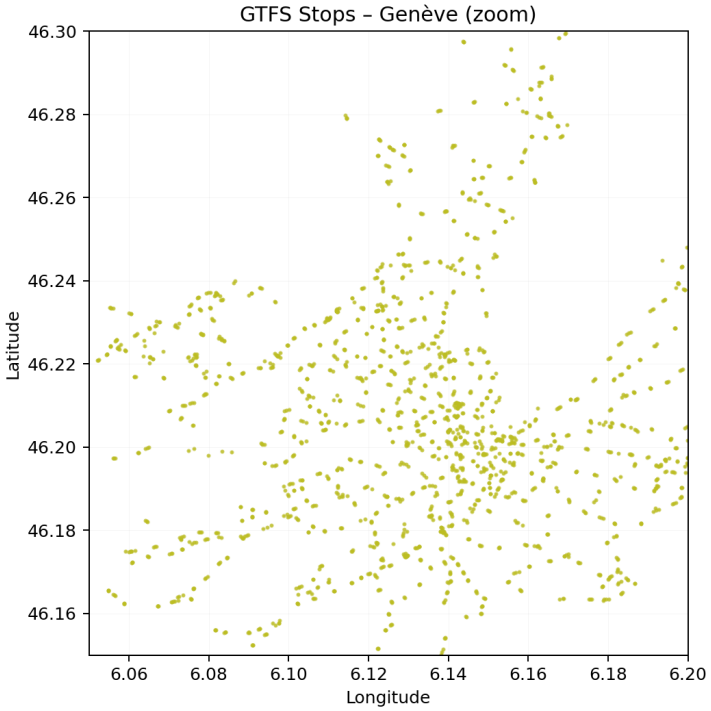


ILLUSTRATION 1.6 – Arrêts GTFS : zoom sur Genève.

Extraits — parsing ciblé HRDF

```
def parse_gleise_lv95_for_sloids(hrdf_path, target_sloids):
    # Associer chaque sloid cible a (UIC, #ref), en un premier passage rapide
    ...

def extract_fplan_directions_for_trips(hrdf_path, target_trip_keys):
    # Pour les trajets cibls, recuperer premier/dernier arret et le nom de ligne
    ...
```

b. Informations exploitées

— **SLOID de quai** et position (WGS84) ;

— **Chaînes directionnelles** *nom* (*Genève* → *Lausanne*) et *UIC* (8501008 → 8501120).

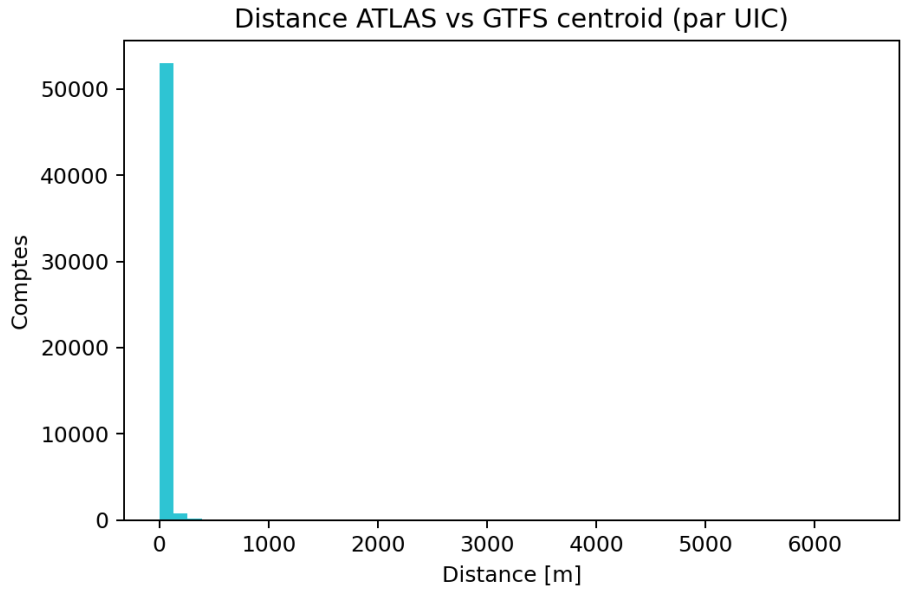


ILLUSTRATION 1.7 – Distribution des distances ATLAS ↔ centroïde GTFS par UIC (mètres).

c. Statistiques clés

Commande

```
 SLOIDs de quai (GLEISE\_LV95)
grep -o "g A ch:1:sloid:[^[:space:]]\\+" data/raw/GLEISE_LV95 | \\
  sed 's/^g A //' | sort -u | wc -l
30935

\# SLOIDs de quai dans BHFART ("G a" = quai, "G A" = gare)
grep -o "G a ch:1:sloid:[^[:space:]]\\+" data/raw/BHFART | sed 's/^G a //' | sort -
    u | wc -l
30935
grep -o "G A ch:1:sloid:[^[:space:]]\\+" data/raw/BHFART | sed 's/^G A //' | sort -
    u | wc -l
31913

\# Toutes occurrences (gare+quai)
grep -o "ch:1:sloid:[^[:space:]]\\+" data/raw/BHFART | sort -u | wc -l
62848
```

Après extraction (`atlas_routes_hrdf.csv`) : 28 757 SLOIDs couverts ; médiane des direc-

tions (noms) par SLOID : **4**. Ces chaînes « nom » et « UIC » enrichissent les correspondances quand un identifiant de direction explicite n'est pas disponible côté OSM.

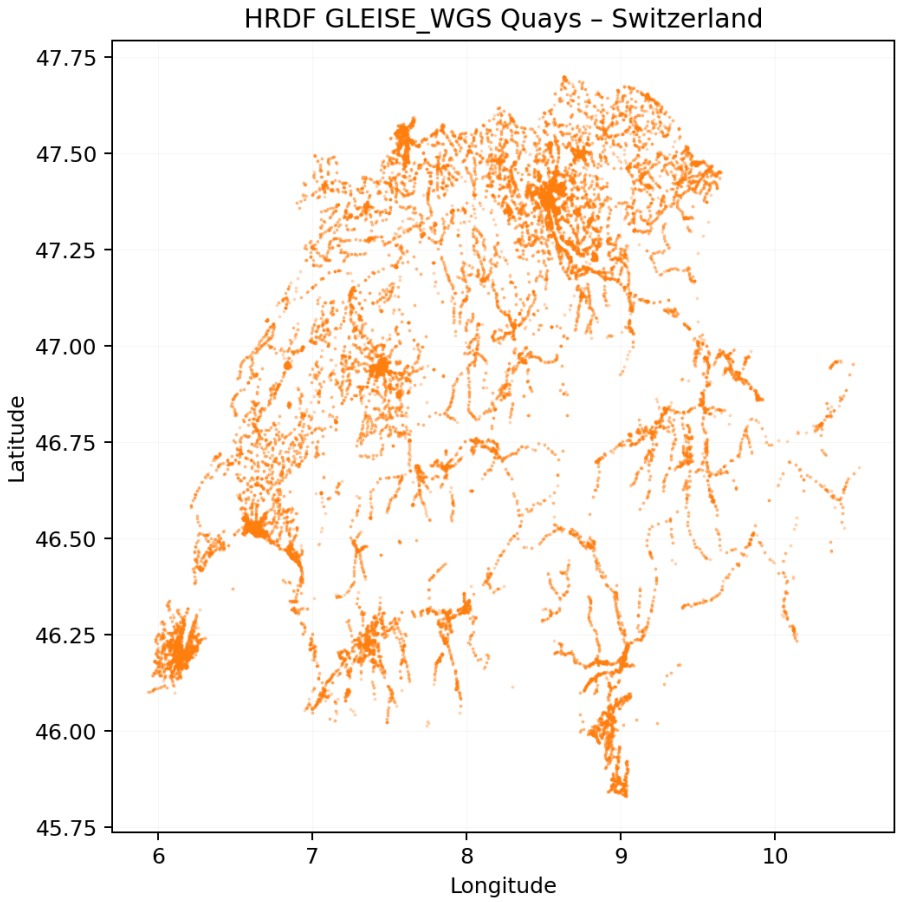


ILLUSTRATION 1.8 – Quais HRDF (`GLEISE_WGS`) – Suisse.

1.4. COMPARAISON GTFS ET HRDF (COUVERTURE SLOID)

— GTFS : 32 248 SLOIDs ; HRDF : 28 757 SLOIDs.

— Intersection : 12 206 ; GTFS seulement : 20 043 ; HRDF seulement : 16 551.

Lien avec la suite. Ces statistiques motivent notre *route matching* (Chapitre 4) qui combine GTFS (identifiants `route_id`/`direction_id`) et HRDF (chaînes *nom*/*UIC*) afin d'optimiser précision et couverture.

1.5. CHAÎNE D'ACQUISITION ET DE PRÉTRAITEMENT (VUE D'ENSEMBLE)

Le pipeline d'ingestion et de préparation des données est centralisé dans le module `get_atlas_data.py`. Il orchestre le téléchargement, le filtrage et l'agrégation des sources amont

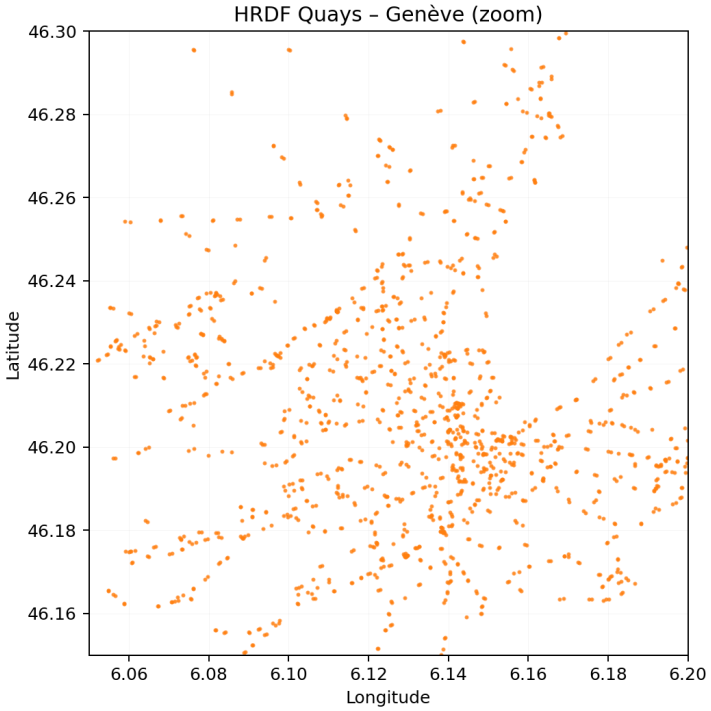


ILLUSTRATION 1.9 – Quais HRDF – zoom Genève.

pour produire des artefacts compacts, déterministes et immédiatement consommables par la chaîne de *matching*. Ses responsabilités principales sont :

1. **ATLAS — acquisition et filtrage** : téléchargement du CSV le plus récent ; restriction à la Suisse (`uicCountryCode = 85`) ; conservation des lignes avec coordonnées WGS84 ; écriture dans `data/raw/stops_ATLAS.csv`.
2. **GTFS — intégration (optimisée)** : extraction du ZIP ; parcours en flux de `stop_times.txt` en deux passes pour (i) identifier les trajets pertinents et les terminus suisses par trajet, (ii) dédupliquer (`stop_id`, `route_id`, `direction_id`) ; jointure minimale avec `trips.txt` et `routes.txt` ; appariement des arrêts GTFS aux SLOIDs ATLAS ; écriture dans `data/processed/atlas_routes_gtfs.csv`.
3. **HRDF — intégration (optimisée)** : extraction du ZIP ; parsing ciblé de `GLEISE_LV95` pour relier chaque `sloid` aux paires (UIC, #ref) nécessaires ; extraction des directions dans `FPLAN` (premier/dernier arrêt) avec résolution des noms via `BAHNHOF` ; écriture dans `data/processed/atlas_routes_hrdf.csv`.

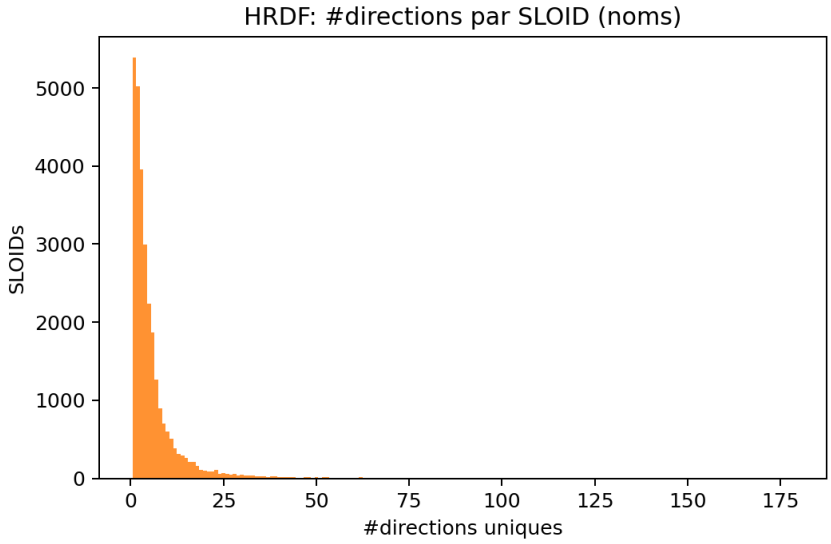


ILLUSTRATION 1.10 – HRDF : distribution du nombre de *directions (noms)* par SLOID.

a. Flux détaillé

ATLAS. Téléchargement depuis *opentransportdata.swiss*, filtrage `uicCountryCode=85`, suppression des coordonnées manquantes. Résultat : référentiel des arrêts (SLOIDs) et attributs essentiels.

GTFS (streaming en deux passes). Plutôt que de matérialiser un `DataFrame stop_times` géant, nous effectuons deux passes en flux :

— *Passe 1* : identification des `trip_id` desservant des arrêts suisses et calcul des terminus suisses par trajet (min/max `stop_sequence`) ; puis chargement *filtré* de `trips.txt` et `routes.txt`.

— *Passe 2* : déduplication de $(\texttt{stop\_id}, \texttt{route\_id}, \texttt{direction\_id})$ et construction de chaînes directionnelles « *Premier* $\rightarrow$ *Dernier* » au niveau `route_id`.

L'appariement `stop_id` $\leftrightarrow$ `sloid` s'appuie sur `uic_number` et `designation` (avec normalisation des références locales « 10000/10001 » $\rightarrow$ « 1/2 »).

HRDF (deux passes ciblées avec gardes rapides). `GLEISE_LV95` est parcouru avec des filtres *très bon marché* (présence de #, préfixe UIC à 7 chiffres, motif « ch :1 :sloid : ») avant toute découpe. *Passe 1* : découverte des paires $(\text{UIC}, \#\text{ref})$ par `sloid`. *Passe 2* : collecte *uniquement* des affectations de trajets dont la paire $(\text{UIC}, \#\text{ref})$ est requise. Les directions sont ensuite

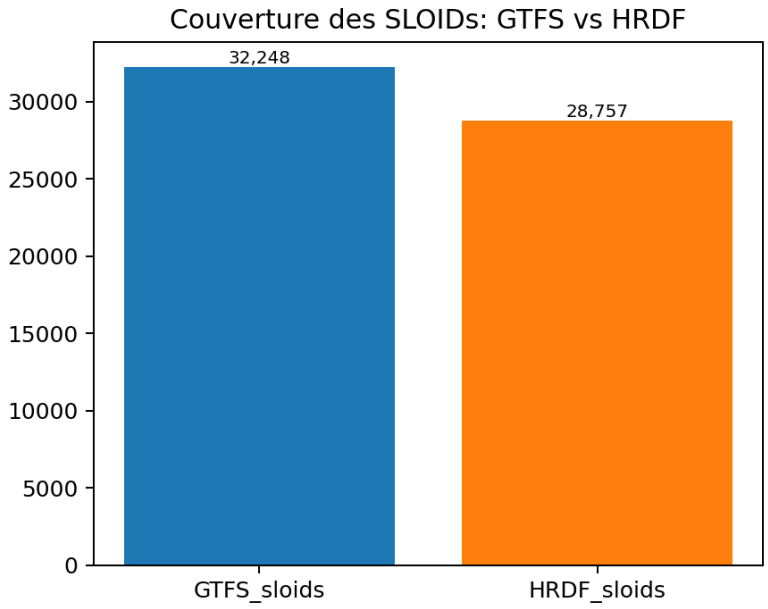


ILLUSTRATION 1.11 – Couverture SLOID : GTFS vs HRDF.

reconstruites via `FPLAN` et les noms d'arrêt de `BAHNHOF`.

1.6. PRINCIPES D'INGÉNIERIE ET D'OPTIMISATION

Nos choix garantissent scalabilité et déterminisme :

— **Streamer plutôt que matérialiser** : passes linéaires avec états compacts au lieu de gros `DataFrames` temporaires.

— **Filtrer tôt et souvent** : réduire très tôt le périmètre (p. ex. `stop_id` suisses commençant par `85`).

— **Parsing ciblé** : gardes substring à coût constant pour éviter `split()` et validations sur des lignes hors-scope.

— **Sorties déterministes** : mêmes résultats que les implémentations antérieures, afin d'isoler les améliorations aux seuls aspects performance/robustesse.

1.7. OPTIMISATIONS ET RÉSULTATS EXPÉRIMENTAUX

a. Ingestion GTFS — streaming en deux passes

Problème. La concaténation/filtration globale de `stop_times` gonfle la mémoire et renchérit les fusions avec `trips/routes`.

Solution. Deux passes en flux (cf. supra), dédupliquant directement

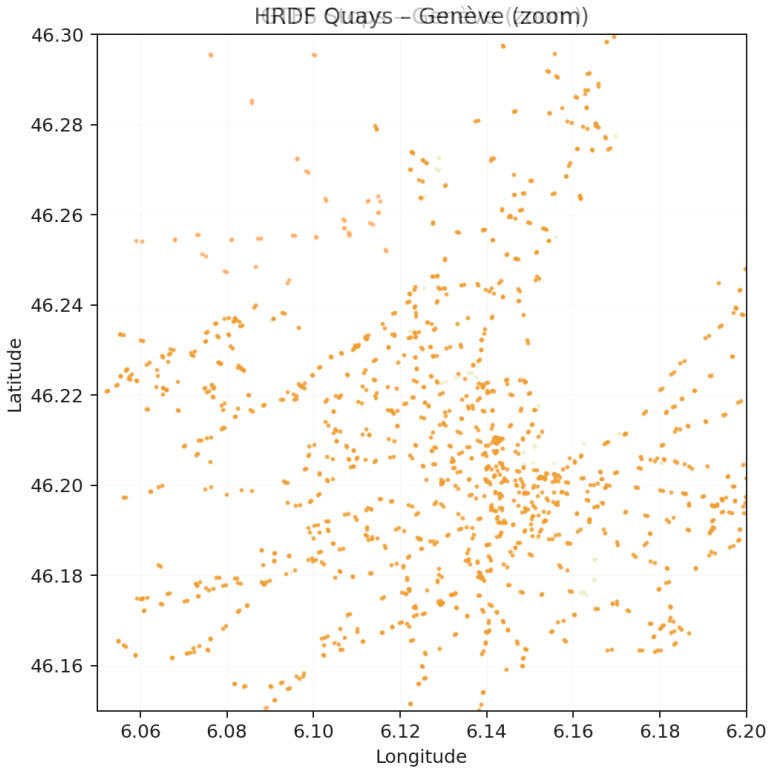


ILLUSTRATION 1.12 – GTFS et HRDF sur la même zone (Genève) — superposition aux mêmes limites spatiales.

(`stop_id`, `route_id`, `direction_id`) et construisant les directions au fil de l'eau.

Impact. Sur un échantillon 2%, la mémoire de crête diminue de plus de moitié à sorties identiques. Plus lent sur un minuscule échantillon (surcoût fixe), le schéma *scale* bien mieux en plein volume car il évite la matérialisation et les *joins* coûteux.

Mode	Temps (s)	Pic RSS Δ (MB)	Lignes
Baseline (2%)	20.17	275.4	8 961
Streaming (2%)	43.28	123.7	8 961

TABLEAU 1.9 – Bench ingestion GTFS sur 2% : fortes économies mémoire, sorties identiques ; en plein volume, le streaming évite les *joins* lourds.

Pourquoi le streaming gagne. Les surcoûts fixes (deux passes, états par *chunk*) pèsent sur petit échantillon. À l'échelle, c'est la matérialisation initiale et les fusions massives qui dominent le coût ; le streaming les supprime.

ILLUSTRATION 1.13 – Vue d'ensemble : acquisition (ATLAS/GTFS/HRDF) → filtrage/streaming → artefacts compacts pour la chaîne de *matching*.

b. Parsing HRDF — deux passes ciblées avec gardes

Problème. `GLEISE_LV95` est volumineux ; découper/valider chaque ligne gaspille du CPU et conserver des mappages larges consomme de la mémoire.

Solution. Gardes substring rapides + deux passes ciblées : (1) découvrir $(\text{UIC}, \#\text{ref})$ par `sloid`, (2) ne conserver que les affectations utiles, puis reconstruire les directions via `FPLAN/BAHNHOF`.

Impact. Sur 2 000 sloids, mêmes résultats, temps nettement réduit et surcoût mémoire négligeable.

Mode	Temps (s)	Pic RSS Δ (MB)
Baseline	26.18	1745.0
Optimisé (deux passes + gardes)	14.54	≈ 0.0

TABLEAU 1.10 – Parsing HRDF : bench sur 2k sloids.

1.8. REPRODUCTIBILITÉ ET EXÉCUTION

Instantané. Sauf mention contraire, toutes les statistiques/cartes reposent sur l'instantané du **11 août 2025** (cf. tête de chapitre).

Scripts. Les scripts de génération des figures et des métriques sont archivés sous `memoire/scripts_used/`. Pour régénérer les artefacts GTFS/HRDF :

Commande

```
python3 get_atlas_data.py
```

Sorties produites.

— `data/raw/stops_ATLAS.csv` — référentiel d'arrêts ATLAS filtré.

— `data/processed/atlas_routes_gtfs.csv` — relations arrêt–route–direction et chaînes directionnelles.

— `data/processed/atlas_routes_hrdf.csv` — lignes `sloid`–direction (noms et UIC) issues d'HRDF.

CHAPITRE 2 : TRANSPORT PUBLIC DANS OSM

La cartographie du transport public dans OpenStreetMap (OSM) a évolué à travers plusieurs schémas. Cette évolution a conduit à la coexistence de diverses combinaisons de balises pour les arrêts de bus, gares ferroviaires, stations de tramway et autres nœuds de transport. De plus, comme OSM est un projet maintenu par une communauté de volontaires, certaines entrées peuvent ne correspondre à aucun schéma précis.

Dans cette section, nous analyserons les différents schémas existants, nous examinerons la requête effectuée (c'est-à-dire quelles données seront utilisées) et, une fois les données obtenues, nous proposerons une vue d'ensemble de l'usage des balises pour les nœuds d'arrêts de transport public dans OSM en Suisse.

2.1. SCHÉMAS DE CARTOGRAPHIE DU TRANSPORT PUBLIC DANS OSM

— **Schéma d'origine (PTv1)** : La méthode la plus ancienne et encore très répandue, qui attribue à chaque arrêt des balises spécifiques au mode concerné. Par exemple, un arrêt de bus est simplement `highway=bus_stop` [5], une gare ferroviaire est `railway=station` (ou `railway=halt` pour des arrêts plus petits), et un arrêt de tramway est `railway=tram_stop`. Ces balises figurent souvent sur un seul nœud représentant l'emplacement où les passagers attendent. PTv1 est largement utilisé encore aujourd'hui [6]. Il est important de noter qu'aucune de ces balises héritées n'a été formellement dépréciée par les propositions plus récentes, ce qui explique qu'elles restent toujours en usage actif.

— **Schéma Oxomoa (années 2010)** : Schéma intermédiaire développé vers 2010 (par l'utilisateur Oxomoa), il introduisait une structure plus aboutie, ressemblant à ce que PTv2 allait proposer plus tard. Ce schéma utilisait des relations de type "route" et des relations de type "stop area" pour regrouper les éléments d'arrêt [6]. Bien qu'il ait influencé la version suivante, ce schéma est désormais historique, même si certains itinéraires plus anciens ($\sim$2010) le suivent encore.

— **Nouveau schéma de transport public (PTv2)** : Approuvé en 2011, PTv2 a introduit un système de balisage plus puissant mais plus complexe [8]. L'idée est de séparer la notion d'arrêt en stop positions (là où le véhicule s'arrête sur la chaussée ou la voie) et

platforms (où les passagers attendent). Dans ce schéma, un arrêt de bus est généralement représenté par *deux* objets reliés :

— un nœud sur la chaussée avec `public_transport=stop_position` (souvent accompagné de `bus=yes` ou `tram=yes`, etc., pour préciser le mode) [7],

— et un nœud (ou une zone) en bord de route portant la balise `public_transport=platform` (en plus d'une balise pour le mode ou d'une balise héritée).

Par exemple, un nœud de plate-forme de bus peut porter `public_transport=platform + bus=yes`, tandis que le nœud correspondant sur la chaussée sera `public_transport=stop_position + bus=yes` [7]. En pratique, les cartographes incluent souvent l'ancienne balise sur l'un de ces objets pour assurer la compatibilité – par exemple, on retrouvera `highway=bus_stop` sur le nœud de la plate-forme, afin qu'il soit reconnu par les outils traditionnels [5].

PTv2 introduit également la notion de relation stop_area (`type=public_transport + public_transport=stop_area`) pour regrouper tous les éléments d'une même station ou d'un même arrêt, et une relation route_master pour regrouper les itinéraires dans les deux sens [8]. Fait notable, la proposition PTv2 n'a pas invalidé ni remplacé les balises existantes, ce qui signifie que les balises PTv1 (telles que `highway=bus_stop`, `railway=station`) coexistent souvent avec les balises PTv2 pour un même arrêt [6]. De nombreuses communautés encouragent à ajouter les balises PTv2 tout en conservant les anciennes pour plus de complétude.

2.2. DIFFÉRENCES DANS L'USAGE DE CLÉS SPÉCIFIQUES

Certains choix de clés varient parmi les cartographes, ce qui peut engendrer des divergences dans la manière de consigner l'information :

— `ref` vs `local_ref` (codes d'arrêt) : De nombreux arrêts de transport public possèdent un code ou identifiant officiel (numéro ou lettre fourni par l'autorité de transport). Les cartographes utilisent tantôt la balise générique `ref=`, tantôt `local_ref=`. La recommandation OSM est : utiliser `ref=` pour le code d'arrêt à l'échelle du réseau (un ID unique dans le système de transport) et `local_ref` si c'est un code/lettre propre à un contexte plus restreint [10].

Par exemple, un arrêt de bus qui a l'ID "3154" dans la base de la ville se balisera `ref=3154`. Et si cet arrêt comporte plusieurs quais, nommés "Bay C" par exemple, on

peut utiliser `local_ref=C` sur le quai concerné. En pratique, la distinction n'est pas toujours respectée : certains mettent tous les codes dans `ref`, d'autres utilisent `local_ref` pour les numéros de quai ou les lettres d'arrêt.

2.3. REQUÊTE OVERPASS TRANSPORT PUBLIC SUISSE

Overpass est un système de requêtage permettant d'extraire des données depuis la base de données OpenStreetMap [11]. Il utilise un langage de requête appelé Overpass Query Language, qui permet de rechercher et filtrer des objets OSM (nœuds, chemins, relations) en fonction de critères spécifiques (tags, zones géographiques, types d'objets, etc.). Pour obtenir les arrêts de transport public en Suisse sur OpenStreetMaps, nous utilisons la requête Overpass suivante :

Requête Overpass

```
[out:xml][timeout:180];
    area["ISO3166-1"="CH"]->.searchArea;
    (
    node(area.searchArea)
    node(area.searchArea)["public_transport"="platform"];
    node(area.searchArea)["public_transport"="stop_position"];
    node(area.searchArea)["public_transport"="station"];
    node(area.searchArea)["public_transport"="halt"];
    node(area.searchArea)["public_transport"="stop"];
    node(area.searchArea)["railway"="tram_stop"];
    node(area.searchArea)["amenity"="ferry_terminal"];
    node(area.searchArea)["amenity"="bus_station"];
    node(area.searchArea)["highway"="bus_stop"];
    node(area.searchArea)["railway"="halt"];
    node(area.searchArea)["railway"="station"];
    node(area.searchArea)["aerialway"="station"];
    );
    out;
    (
      relation(bn)[type=route];
    );
    out meta;
```

Cette requête commence par définir la zone d'intérêt, qui correspond à la Suisse, identifiée par le code ISO3166-1 `CH`. Ensuite, elle sélectionne différents types de nœuds correspondant aux infrastructures de transport public. Cette requete inclut des arrets de bus, tram, les terminaux de ferries, les stations de remontées mécaniques, etc.

Enfin, la requête extrait également les relations de type `route` associées aux nœuds obtenus. Cette information est pertinente, car elle permet de lier les arrêts à leurs itinéraires respectifs, ce qui facilitera les correspondances avec d'autres sources de données, comme les données Atlas. On vera plus de détails sur les donnés des routes dans la section 1.5.

2.4. Aperçu des "tags" des nœuds de transport public dans OSM en Suisse

Une fois qu'on a les nœuds dans la requête ci-dessus, nous avons analysé quels tags est-ce que ceux nœuds ont. D'abord on va montrer quelques exemples de noeds pour le lecteur :

OSM Nœud : Grand-Mont

```
<node id="2368323780" lat="46.5627599" lon="6.6343369">
   <tag k="bus" v="yes"/>
   <tag k="highway" v="bus_stop"/>
   <tag k="local_ref" v="D"/>
   <tag k="name" v="Grand-Mont"/>
   <tag k="network" v="Mobilis"/>
   <tag k="operator" v="TL"/>
   <tag k="public_transport" v="stop_position"/>
   <tag k="tactile_paving" v="no"/>
   <tag k="trolleybus" v="yes"/>
   <tag k="uic_name" v="Le Mont-sur-L., Grand-Mont"/>
   <tag k="uic_ref" v="8504177"/>
 </node>
```

OSM Nœud sans nom

```
<node id="2368860496" lat="46.4418646" lon="6.9764107">
   <tag k="aerialway" v="station"/>
 </node>
```

OSM Nœud : Interlaken Ost

```
</node>
 <node id="2388274179" lat="46.6910098" lon="7.8697428">
   <tag k="name" v="Interlaken Ost"/>
   <tag k="public_transport" v="stop_position"/>
   <tag k="railway" v="stop"/>
   <tag k="ref" v="7"/>
   <tag k="train" v="yes"/>
 </node>
```

Comme vous pouvez le voir dans les exemples ci-dessus, chaque nœud contient des tags différents. Voici quelques statistiques pour avoir une vision générale.

- Nombre total de nœuds : 60 635
- Nombre total de nœuds avec `public_transport` == `platform` : 24 548
  - Parmi ceux-ci avec `uic_ref` : 22 986
  - Nœuds de plateforme avec une position d'arrêt correspondante (même `uic_ref`) : 13 571
  - Nœuds de plateforme avec `uic_ref` mais sans position d'arrêt correspondante : 9 415
- Nombre total de nœuds avec `public_transport` == `stop_position` : 30 018
  - Parmi ceux-ci avec `uic_ref` : 28 199
- Nœuds avec toutes les balises (`uic_ref`, `local_ref`, `name`, `network`, `operator`, `uic_name`) : 3 875
- Nœuds avec `uic_ref` : 55 166
  - Parmi ceux-ci, avec `ref` : 2 796
  - Parmi ceux-ci, avec `local_ref` : 4 314
  - Parmi ceux-ci, avec `ref` et `local_ref` : 307
  - Parmi ceux-ci avec `name` : 55 147
  - Parmi ceux-ci avec `network` : 40 576
  - Parmi ceux-ci avec `operator` : 53 322
  - Parmi ceux-ci avec `uic_name` : 55 042
- Nœuds sans `uic_ref` : 5 469

  - Parmi ceux-ci, avec `ref` : 200
  - Parmi ceux-ci, avec `local_ref` : 288
  - Parmi ceux-ci, avec `ref` et `local_ref` : 13
  - Parmi ceux-ci avec `name` : 4 339
  - Parmi ceux-ci avec `network` : 758
  - Parmi ceux-ci avec `operator` : 1 542
  - Parmi ceux-ci avec `uic_name` : 86
- Nombre total de nœuds sans aucune des balises `uic_ref`, `ref`, `local_ref`, `network`, `operator`, `uic_name` : 1 084
- Nœuds non assignés avec `aerialway=station` : 817

2.5. Aperçu des itinéraires de transport public dans OSM en Suisse

Comme mentionné dans le chapitre 1, nous nous intéressons également aux itinéraires, car ils peuvent nous aider à identifier des correspondances. Cela est particulièrement utile lorsqu'il existe deux arrêts pour une même station, mais pour des itinéraires empruntant des directions opposées, ou lorsque des arrêts de bus et de tram sont situés à proximité.

Voici quelques statistiques essentielles :

- Total d'itinéraires uniques : 7 525
- Total de connexions entre nœuds et itinéraires : 139 084
- Nombre de nœuds desservant au moins un itinéraire : 51 182
- Nombre moyen d'itinéraires par nœud : 2,72

a. Les 5 nœuds de transport public les plus fréquentés

Top 5 des nœuds avec plus des itinéraires

1. Node ID: 5962551000
   Nom : Bus Station Zürich
   Itinéraires desservis : 65
   Type de nœud : stop_position

2. Node ID: 3309712200
   Nom : Sursee Bahnhof
   Itinéraires desservis : 43
   Type de nœud : stop_position
   Référence UIC : 8502998

3. Node ID: 984028248
   Nom : Stein
   Itinéraires desservis : 40
   Type de nœud : stop_position
   Référence UIC : 8580638

4. Node ID: 960890428
   Nom : Genève - Gare Routière
   Itinéraires desservis : 37
   Type de nœud : stop_position

5. Node ID: 984002736 / 1236383343
   Nom : Lugano Centrale / Zürich Hauptbahnhof
   Itinéraires desservis : 37
   Type de nœud : stop_position / station
   Référence UIC : 8505550 / 8503000

b. Analyse des directions des itinéraires

— Direction 0 (généralement sortante) : 70 531 connexions

— Direction 1 (généralement entrante) : 38 822 connexions

— Direction inconnue : 29 731 connexions

c. Top 5 des itinéraires avec le plus d'arrêts

— Bus 120 : Engelburg → St. Gallen → Eggersriet → Heiden : 174 arrêts

— Bus 120 : Heiden → Eggersriet → St. Gallen → Engelburg : 174 arrêts

— Bus 722 : Weinfelden → Hosenruck → Wil SG : 150 arrêts

— Bus 722 : Wil SG → Hosenruck → Weinfelden : 150 arrêts

— Bus 507 : Lostorf → Olten → Egerkingen : 138 arrêts

CHAPITRE 3 : CORRESPONDANCE AVEC LES DONNÉES ATLAS-OSM

Le processus de correspondance entre les données ATLAS et OSM a été conçu pour identifier de manière précise et systématique les arrêts correspondants dans ces deux ensembles de données. Il se déroule en plusieurs étapes, chacune reposant sur des critères spécifiques pour maximiser la fiabilité des correspondances tout en minimisant les erreurs. Les étapes incluent une correspondance exacte basée sur les identifiants, une correspondance par nom et une correspondance par distance avec plusieurs sous-étapes. Voici une description détaillée de chaque étape.

3.1. CORRESPONDANCE EXACTE

La première étape, appelée correspondance exacte, utilise l'identifiant UIC. Dans les données ATLAS, cet identifiant est représenté par la colonne `'number'`, tandis que dans OSM, il correspond à la balise `'uic_ref'`. Une entrée ATLAS est appariée à un nœud OSM si son `'number'` est identique au `'uic_ref'` du nœud OSM.

Des situations complexes peuvent survenir lorsque plusieurs entrées ATLAS partagent le même `'number'` (par exemple, plusieurs quais d'une même gare) ou lorsque plusieurs nœuds OSM possèdent le même `'uic_ref'`. Pour résoudre ces cas, les règles suivantes sont appliquées :

1. **Cas 1 : Plusieurs entrées ATLAS, un seul nœud OSM** Si plusieurs entrées ATLAS partagent le même `'number'` et qu'un seul nœud OSM possède ce `'uic_ref'`, toutes ces entrées ATLAS sont appariées à ce nœud OSM unique.
2. **Cas 2 : Une entrée ATLAS, plusieurs nœuds OSM** Si une seule entrée ATLAS a un `'number'` donné et que plusieurs nœuds OSM partagent ce `'uic_ref'`, tous ces nœuds OSM sont appariées à cette entrée ATLAS unique.
3. **Cas 3 : Plusieurs entrées ATLAS et plusieurs nœuds OSM** Lorsque plusieurs entrées ATLAS et nœuds OSM partagent le même `'number'`/`'uic_ref'`, une correspondance plus fine est réalisée en comparant la `'designation'` de l'entrée ATLAS (par exemple, le code du quai) avec la balise `'local_ref'` du nœud OSM. Une correspondance est établie si ces valeurs sont identiques.

Cette méthode a permis d'identifier 21994 correspondances exactes.

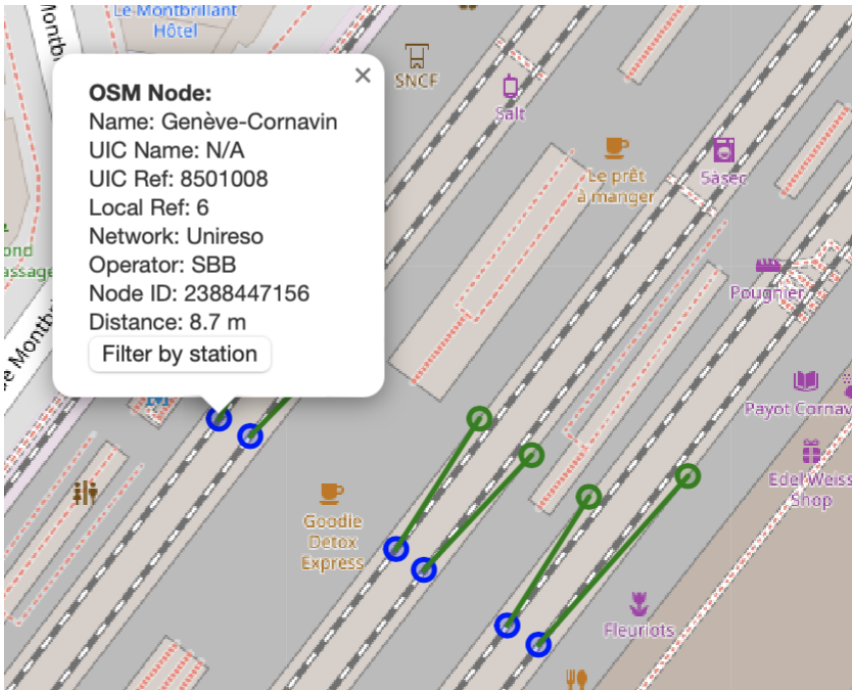


ILLUSTRATION 3.1 – Correspondances exactes à la gare de Genève-Cornavin. Les détails du nœud OSM de la voie 6 sont visibles sur l'image.

3.2. CORRESPONDANCE PAR NOM

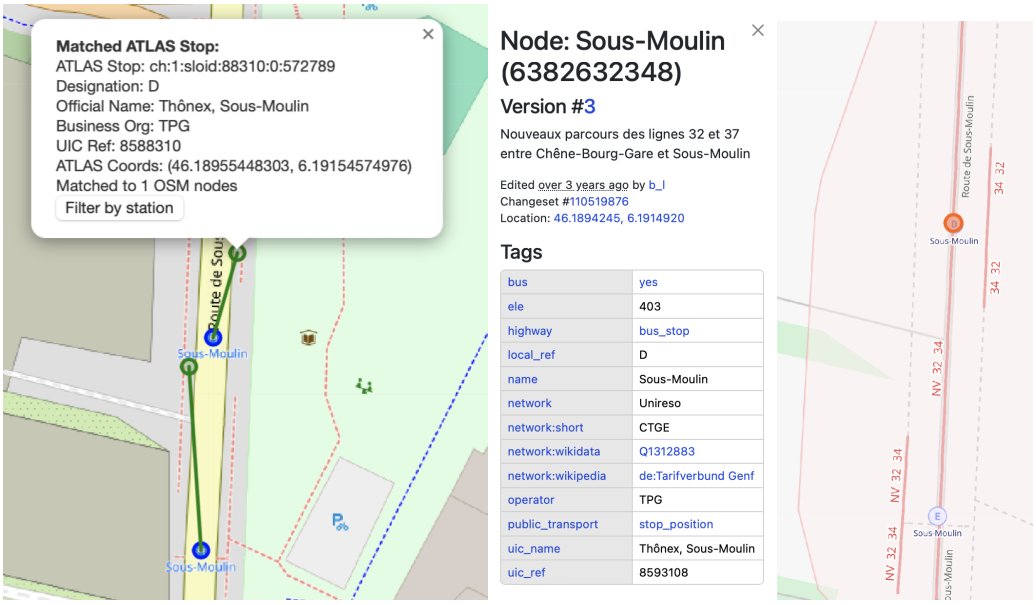
Pour les entrées ATLAS non appariées lors de l'étape précédente, une correspondance basée sur le nom est appliquée. Cette étape compare le nom officiel des arrêts, indiqué dans la colonne `'designationOfficial'` des données ATLAS, avec plusieurs balises de nom dans OSM : `'name'`, `'uic_name'` et `'gtfs:name'`.

La règle principale établit une correspondance si le `'designationOfficial'` correspond exactement à l'une de ces balises OSM. Cependant, si plusieurs nœuds OSM présentent le même nom correspondant, un critère supplémentaire est utilisé : la balise `'local_ref'` du nœud OSM est comparée à la `'designation'` de l'entrée ATLAS. Une correspondance est confirmée si ces valeurs sont identiques (en ignorant la casse).

Cette approche a permis d'ajouter 339 correspondances supplémentaires.

3.3. CORRESPONDANCE PAR DISTANCE

Pour les entrées ATLAS restantes, une correspondance basée sur la proximité géographique est mise en œuvre. Cette étape se divise en trois sous-étapes distinctes, chacune avec des critères spécifiques pour garantir des appariements fiables.



(A) Correspondances par nom. (B) Capture d'écran d'un nœud OSM..

ILLUSTRATION 3.2 – Pour l'arrêt "Thônex, Sous-Moulin, D", on peut voir que, malgré une référence UIC différente, il est possible d'établir des correspondances grâce au nom.

a. Étape 1 : Correspondance de groupe basée sur la proximité

Les entrées ATLAS et OSM sont regroupés selon les paires d'identifiants suivantes :

1. `'number'` (ATLAS) et `'uic_ref'` (OSM).
2. `'designationOfficial'` (ATLAS) et `'uic_name'` (OSM).
3. `'designationOfficial'` (ATLAS) et `'name'` (OSM).

Dans chaque groupe où le nombre d'entrées ATLAS est égal au nombre de nœuds OSM, une correspondance est tentée en associant chaque entrée ATLAS au nœud OSM le plus proche, à condition que cette association soit cohérente (c'est-à-dire que chaque nœud OSM soit également le plus proche de l'entrée ATLAS qui lui est attribuée). Cette méthode nous a permis de réaliser 8 902 correspondances supplémentaires.

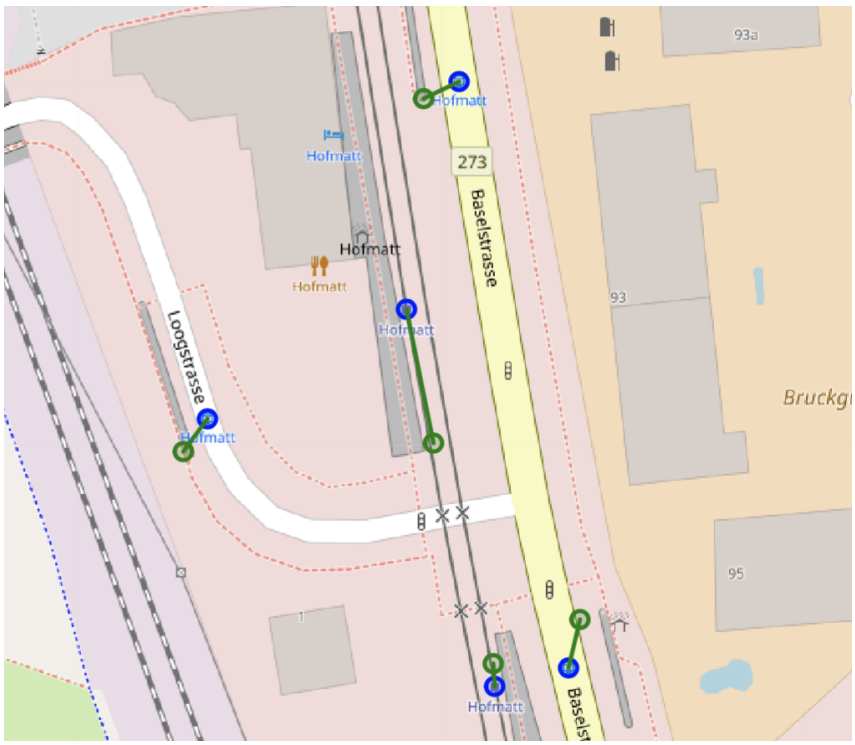


ILLUSTRATION 3.3 – Correspondances pour les arrêts de Münchenstein, Hofmatt. Malgré les divergences de `uic_ref` et le manque de références locales, nous avons réussi à établir des correspondances grâce à la correspondance de groupe basée sur les distances.

TABLEAU 3.1 – Données ATLAS pour les arrêts de Münchenstein, Hofmatt

sloid	number	designation	designationOfficial
ch :1 :sloid :95 :1 :6	8500095		Münchenstein, Hofmatt
ch :1 :sloid :95 :1 :5	8500095		Münchenstein, Hofmatt
ch :1 :sloid :95 :1 :3	8500095		Münchenstein, Hofmatt
ch :1 :sloid :95 :1 :2	8500095		Münchenstein, Hofmatt
ch :1 :sloid :95 :1 :1	8500095		Münchenstein, Hofmatt

TABLEAU 3.2 – Données OSM pour les arrêts de Münchenstein, Hofmatt

node_id	uic_ref	uic_name	transport_type
6457499611	8578185	Münchenstein, Hofmatt	bus
299126238	8500095	Münchenstein, Hofmatt	tram
983964446	8578185	Münchenstein, Hofmatt	bus
1435404358	8500095	Münchenstein, Hofmatt	tram
3858822225	8578185	Münchenstein, Hofmatt	bus

b. Étape 2 : Correspondance par référence locale dans un rayon de 50 mètres

Cette sous-étape recherche, pour chaque entrée ATLAS non appariée, un nœud OSM situé à moins de 50 mètres dont la balise `local_ref` correspond exactement à la `designation` de l'entrée ATLAS (en ignorant la casse).

À Zürich HB, dans ATLAS, la `UIC_ref` est égale à 8503000 pour tous les arrêts, tandis que dans OSM, certains arrêts ont une `UIC_ref` de 8516144.

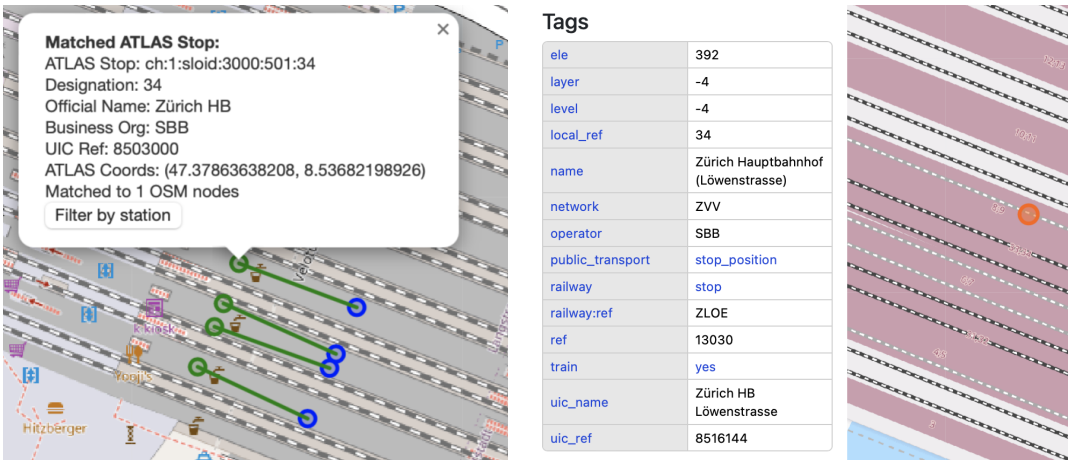


ILLUSTRATION 3.4 – Correspondances à Zürich HB grâce à l'étape 2.

Cette méthode nous a permis de réaliser 127 correspondances supplémentaires.

c. Étape 3 : Correspondance basée sur la proximité avec critères relatifs

Pour les entrées toujours non appariées, tous les nœuds OSM situés à moins de 50 mètres sont examinés :

— a) Si un seul nœud OSM se trouve dans ce rayon, il est apparié à l'entrée ATLAS.

— Si plusieurs nœuds OSM sont présents, l'appariement est effectué avec le nœud le plus proche uniquement si :

1. b) Le deuxième nœud le plus proche est à au moins 10 mètres.
2. La distance au deuxième nœud le plus proche est au moins 4 fois supérieure à celle du nœud le plus proche.

Nous avons réussi à établir 2 233 correspondances avec l'option a) et 1 983 correspondances avec l'option b). Cette méthode est utile pour les cas où il y a des nœuds isolés, comme des télésièges.

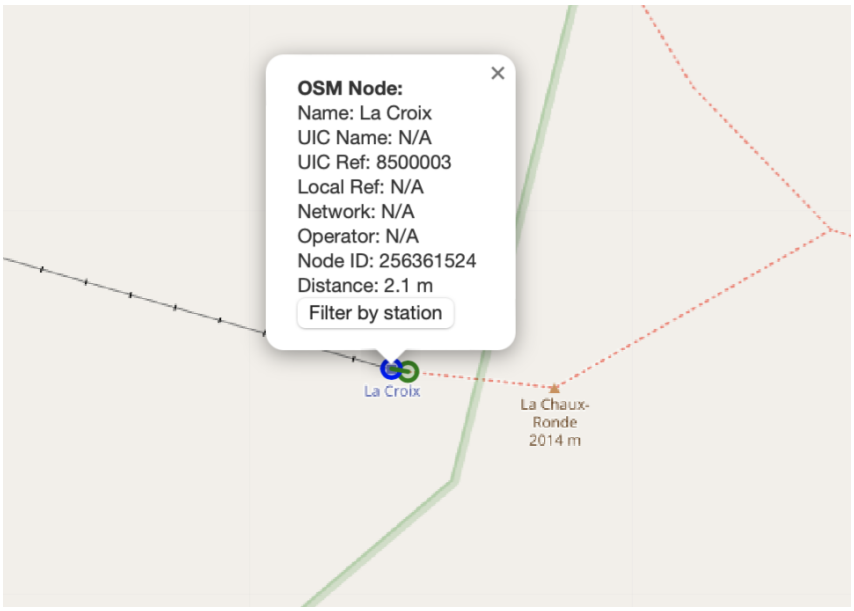


ILLUSTRATION 3.5 – Correspondance par distance étape 3

3.4. RÉSULTATS ACTUELS

Parmi les 56.128 arrêts ATLAS que nous avons considérés jusqu'à présent, le processus de correspondance a permis d'identifier un total de 33.747 correspondances entre les données ATLAS et OSM. Après ces étapes, 22.380 entrées ATLAS restent non appariées, et 34.522 nœuds OSM restent inutilisés. Parmi ces nœuds OSM inutilisés, 27.759 sont associés à au moins une route, 28.068 possèdent une référence UIC, et 1.908 ont une référence locale (`local_ref`).

CHAPITRE 4 : CORRESPONDANCE PAR ITINÉRAIRE

CHAPITRE 5 : ANALYSE DES RÉSULTATS

5.1. DISTANCES LES PLUS IMPORTANTES

Le top 5 des correspondances avec les distances les plus importantes sont présentés dans le tableau suivant :

Distance (m)	SLOID	OSM Node	OSM Name
3226.69	ch :1 :sloid :1707 :0 :1001	983835813	Villars-sur-Ollon, La Roche
3226.63	ch :1 :sloid :1707 :0 :1002	983835813	Villars-sur-Ollon, La Roche
2487.66	ch :1 :sloid :30100 :0 :290588	250138382	Chassoure
1879.04	ch :1 :sloid :30169 :0 :346485	329715617	Les Violettes
1451.90	ch :1 :sloid :277 :0 :693607	243721801	Espel

TABLEAU 5.1 – Top 5 des correspondances avec les distances les plus importantes

Une fois le processus de correspondance terminé, on procédera à une analyse approfondie des erreurs dans ce cas et on essaiera de trouver une manière systématique de les résoudre.

5.2. STATISTIQUES DE DISTANCE

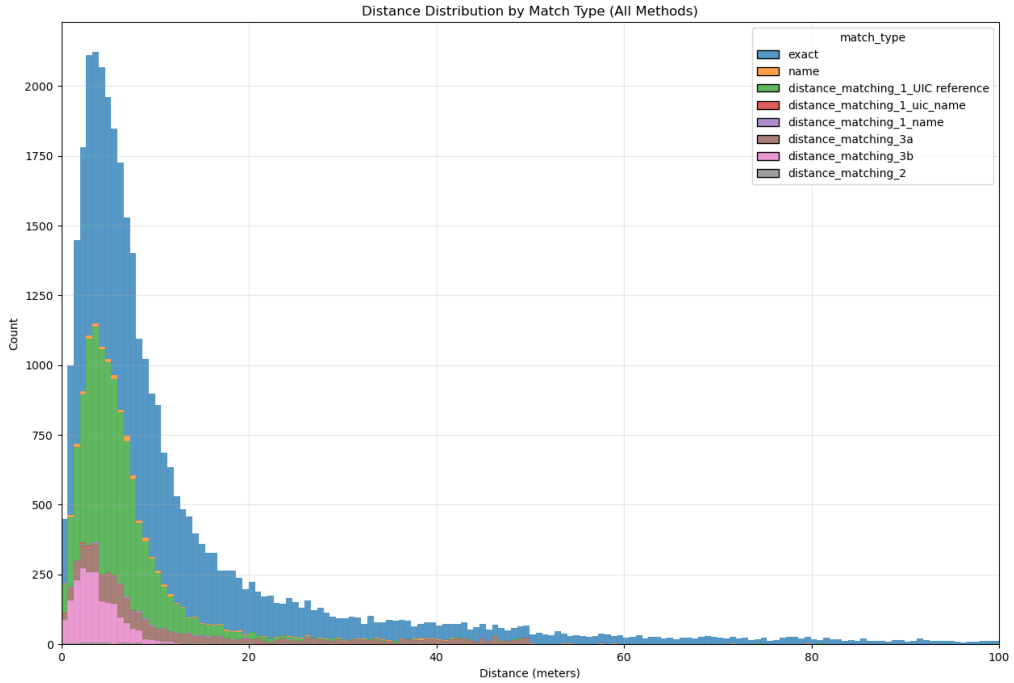


ILLUSTRATION 5.1 – Distributions des distances pour chaque méthode de correspondance

Comme on peut le voir dans le graphique, à une coupe de 100 mètres, il y a une longue traîne de correspondances qu'il faut corriger. De plus, la médiane pour les correspondances exactes est très importante à 22 mètres.

Voici les statistiques :

- **DISTANCE**
  - Moyenne : 7,94 m
  - Médiane : 5,37 m
- **EXACT**
  - Moyenne : 22,62 m
  - Médiane : 9,40 m
- **NAME**
  - Moyenne : 36,64 m
  - Médiane : 10,25 m

CHAPITRE 7 : PROBLÈMES DETECTÉS

CHAPITRE 7 : BASE DE DONNÉES

CHAPITRE 8 : BACKEND

CHAPITRE 9 : FRONTEND

CHAPITRE 10 : ARCHITECTURE DU RENDU CARTOGRAPHIQUE ET OPTIMISATIONS DE PERFORMANCE

Ce chapitre résume l'architecture de la carte interactive, le flux de données client–serveur et les optimisations qui rendent la navigation réactive avec des volumes de données réalistes. Il expose succinctement les choix de conception et indique où des visuels renforceraient l'explication.

10.1. VUE D'ENSEMBLE DU SYSTÈME

La carte (Leaflet) rend quatre classes d'entités : paires appariées (ATLAS–OSM), arrêts ATLAS non appariés, nœuds OSM non appariés et agrégats de stations. Le client ne sollicite que ce qui est nécessaire pour dessiner les marqueurs dans la fenêtre d'affichage courante ; les détails riches des info-bulles sont chargés à la demande lorsque l'utilisateur interagit.

Les données sont persistées à l'aide d'une table cœur normalisée `stops` (coordonnées, identifiants, types d'arrêt et d'appariement, distance) et de tables de détails `atlas_stops` et `osm_nodes` (noms, opérateurs, lignes, notes). Cette séparation permet de dessiner la carte à partir de `stops` uniquement pour la navigation, en reportant les métadonnées lourdes à des chargements à la demande.

Illustration (suggestion). Un petit diagramme de séquence : l'utilisateur se déplace/zoome → le client appelle `/api/data` (minimal) → les marqueurs apparaissent ; au clic → le client appelle `/api/stop_popup` → les détails de l'info-bulle s'affichent.

10.2. ARCHITECTURE DU BACKEND

a. Points de terminaison

— `/api/data` : renvoie un ensemble minimal d'enregistrements pour une fenêtre d'affichage et des filtres donnés (identifiants, attributs grossiers, coordonnées). Aucun itinéraire/aucune note n'est inclus(e).

— `/api/stop_popup` : renvoie des détails enrichis pour un arrêt unique à la demande (noms, opérateurs, lignes, notes, et contexte de multi-appariement le cas échéant).

— `/api/top_matches`, `/api/global_stats` : rapports et synthèses d'en-tête.

— La page Problèmes utilise également `/api/data` pour une petite fenêtre de contexte spatial (§10.5).

b. Filtrage par fenêtre d'affichage (sargable)

Pour sélectionner les entités visibles, le serveur utilise un prédicat *OR-of-ANDs* sargable au lieu de `COALESCE`, ce qui permet d'exploiter les index B-Tree sur les paires de coordonnées ATLAS et OSM :

```
(atlas_lat IS NOT NULL AND atlas_lon IS NOT NULL AND
 atlas_lat BETWEEN :min_lat AND :max_lat AND
 atlas_lon BETWEEN :min_lon AND :max_lon)
OR
(atlas_lat IS NULL AND atlas_lon IS NULL AND
 osm_lat IS NOT NULL AND osm_lon IS NOT NULL AND
 osm_lat BETWEEN :min_lat AND :max_lat AND
 osm_lon BETWEEN :min_lon AND :max_lon)
```

c. Politique de charge utile

Charge utile de navigation minimale uniquement. L'endpoint `/api/data` renvoie toujours une structure JSON compacte suffisante pour rendre les marqueurs (identifiants, types, coordonnées, distance, identifiants sélectionnés). N'y figurent jamais les itinéraires, les notes ou le contenu des tables liées ; les info-bulles s'appuient entièrement sur `/api/stop_popup`. Cela réduit la taille des charges utiles et accélère l'analyse à tous les niveaux de zoom.

Le serveur omet intentionnellement les structures coûteuses d'agrégation des multi-appariements dans `/api/data`. Les éléments visuels à fort zoom (p. ex. lignes de liaison) sont dérivés côté client à partir des coordonnées ligne par ligne lorsque disponibles.

d. Indexation de la base de données

Les index qui prennent en charge les schémas d'accès courants incluent : `stops(atlas_lat, atlas_lon)`, `stops(osm_lat, osm_lon)` (fenêtre d'affichage), `stops(stop_type, match_type)` (filtres), et `atlas_stops(atlas_business_org_abbr)` (filtres opérateur).

10.3. ARCHITECTURE DU FRONTEND

a. Cycle de vie des requêtes et limitation du débit

Pour maintenir l'interface réactive lors des déplacements/zooms :

— **Anti-rebond** (debounce) de `moveend`/`zoomend` à $\sim 320\,\text{ms}$ avant de récupérer les données de la nouvelle fenêtre d'affichage.

— **Annulation des requêtes en cours** lorsqu'une nouvelle démarre ; ignorer les réponses obsolètes.

— **Plafond à zoom intermédiaire** : récupérer jusqu'à 500 entités par requête pour garder de petites charges utiles.

— **Sans plafond à fort zoom** : une fois le seuil atteint plus deux niveaux de zoom, le client omet la limite et le serveur renvoie *tous* les marqueurs dans la fenêtre d'affichage.

— **Exception « petit ensemble » à faible zoom** : aux faibles zooms (en dessous du seuil d'affichage des marqueurs), le client effectue une requête de sondage avec un plafond plus petit (≤ 250). Si l'ensemble filtré contient ≤ 250 entités, elles sont rendues même à faible zoom. Sinon, une bannière s'affiche et le rendu est différé jusqu'à ce que l'utilisateur zoome.

Illustration (suggestion). Un diagramme temporel montrant des déplacements rapides, la fenêtre d'anti-rebond de 320 ms et l'annulation des requêtes précédentes.

b. Seuils de zoom et politique visuelle

Nous alignons le travail visuel sur l'intention :

— **Masquer les marqueurs** en dessous de $z < 13$; afficher une petite bannière invitant l'utilisateur à zoomer. Le texte de la bannière est *« Zoomez pour voir tous les marqueurs d'arrêt »* et reste visible pendant *deux* niveaux de zoom après la première apparition des marqueurs afin d'indiquer que tous les marqueurs ne sont peut-être pas encore affichés.

— **Masquer les lignes de liaison** en dessous de $z < 14$; ne tracer les lignes ATLAS $\rightarrow$OSM qu'aux niveaux de zoom plus élevés.

c. Rendu des marqueurs : Canvas d'abord, lettres en dernier

Les icônes basées sur le DOM (`L.marker` avec `divIcon`) ont un coût élevé lorsqu'elles sont nombreuses. Nous :

- Préférer les **cercles Canvas** (`L.circleMarker`) aux faibles et moyens niveaux de zoom et activer l'option de carte `preferCanvas` de Leaflet.
- Rendre les **icônes lettrées** (`D`, `P`, `S`) uniquement à partir de $z \geq 18$. Aux niveaux de zoom inférieurs, les mêmes nœuds utilisent des cercles légers.
- **Mettre en cache** les instances identiques de `divIcon` par clé (type+couleur+libellé) pour éviter de recréer du DOM.
- **Ajouter par lots** les marqueurs aux couches pour éviter de longs blocages du thread principal lors d'inserts massifs.
- Appliquer un léger **motif de décalage** en cas de superposition exacte des coordonnées afin d'éviter l'occlusion visuelle sans recourir à un regroupement complet.

Illustration (suggestion). Une figure en trois panneaux de la même zone à $z = 12$ (pas de marqueurs, bannière), $z = 15$ (cercles Canvas uniquement) et $z = 18$ (marqueurs lettrés pour le sous-ensemble qui en a besoin).

10.4. Paires appariées et polylignes

À fort zoom, nous traçons des lignes de liaison entre les coordonnées ATLAS et OSM pour les entités appariées lorsque les deux extrémités sont visibles. Les lignes sont omises aux faibles zooms pour réduire l'encombrement et le coût de rendu. Comme `/api/data` est strictement minimal, cette logique s'applique directement aux coordonnées de chaque ligne ; un contexte relationnel plus riche, si nécessaire, est récupéré via `/api/stop_popup`.

10.5. Récupération du contexte sur la page Problèmes

La page Problèmes affiche des *marqueurs de contexte* autour du problème courant à l'aide d'une petite boîte englobante ($\pm 0.02^\circ$, $\approx 2\,\text{km}$), de résultats plafonnés (150–200) et de la même politique de rendu (Canvas d'abord, lignes uniquement à fort zoom). Les info-bulles sont chargées paresseusement via `/api/stop_popup`, et les marqueurs sont ajoutés par lots. La carte active aussi `preferCanvas` pour un dessin vectoriel rapide.

10.6. CONSIDÉRATIONS D'EXPÉRIENCE UTILISATEUR

a. Échantillonnage et limites

Aux zooms intermédiaires, le client plafonne les résultats (≤ 500) et le serveur renvoie un sous-ensemble. À fort zoom (seuil + deux niveaux), le client omet la limite et *tous* les marqueurs qui correspondent dans la fenêtre d'affichage sont renvoyés. L'interface communique le contexte via la bannière de zoom et un rendu incrémental rapide. À faible zoom, le client émet une requête de sondage avec un plafond plus petit ; si le nombre de résultats est faible (≤ 250), ils sont rendus immédiatement. Sinon, la bannière s'affiche et le rendu complet est différé jusqu'à ce que l'utilisateur zoome. Une amélioration facultative consiste en une petite note du type *« Affichage de N sur M. Zoomez pour en voir davantage. »*

b. Statistiques d'en-tête

Les statistiques globales se rafraîchissent en parallèle des chargements de données et s'appuient sur des agrégations optimisées afin de ne pas dégrader la navigation interactive.

10.7. IMPACT SUR LES PERFORMANCES

Gains qualitatifs observés sur l'ensemble des jeux de données :

- **Serveur** : CPU plus faible grâce aux filtres sargables, à la sérialisation de charges utiles plus petites et aux requêtes anti-rebond/annulées.
- **Réseau** : JSON systématiquement plus petit (strictement minimal), analyse plus rapide.
- **Client** : moins de nœuds DOM (lettres uniquement à $z \geq 18$), rendu Canvas d'abord, icônes mises en cache et insertions par lots réduisent le travail du thread principal.

Illustration (suggestion). Un histogramme comparant les tailles de charge utile (`/api/data` minimal vs `/api/stop_popup` à la demande) et une courbe du temps par image avant/après « Canvas d'abord » + insertions par lots.

10.8. RÉSUMÉ

Le système atteint des performances interactives en associant une charge utile de navigation **strictement minimale** à des **détails d'info-bulle différés**, un filtrage de fenêtre d'affichage sargable, des requêtes anti-rebond/annulables et un rendu **Canvas d'abord**. Les icônes DOM

lettrées sont différées à des niveaux de zoom très élevés et réutilisées via la mise en cache ; les marqueurs sont insérés par lots pour maintenir la réactivité du thread principal. Ensemble, ces mesures réduisent le travail côté serveur, le coût réseau et la charge de rendu côté client tout en préservant la clarté lors d'une inspection détaillée.

CHAPITRE 11 : SYSTÈME D'AUTHENTIFICATION SÉCURISÉ

11.1. OBJECTIFS

Nous avons conçu et mis en œuvre un système d'authentification moderne qui priorise la confidentialité, l'intégrité et la disponibilité. Il est aligné sur les bonnes pratiques actuelles (2025) et est prêt pour une revue de sécurité.

11.2. APERÇU DE L'ARCHITECTURE

L'application utilise Flask avec un schéma d'authentification dédié lié à une base de données MySQL distincte (`auth_db`). Les données analytiques principales demeurent dans `stops_db`. Cette séparation réduit le rayon d'impact et garantit que la réimportation des données analytiques ne touche jamais les identifiants des utilisateurs.

a. Composants

— **Stockage des mots de passe** : Argon2id (argon2-cffi), à forte consommation mémoire, salé, avec des paramètres spécifiques à chaque hachage.
— **Sessions** : Flask-Login avec cookies sécurisés (HttpOnly, SameSite=Lax ; indicateur Secure en production).
— **CSRF** : Protection CSRF de Flask-WTF pour tous les formulaires POST et les points de terminaison concernés.
— **Limitation de débit** : Les routes de connexion et d'inscription sont limitées (Flask-Limiter) afin d'atténuer la force brute.
— **Sécurité du transport** : Flask-Talisman fournit les en-têtes de sécurité ; l'application stricte de HTTPS est configurable via les variables d'environnement.
— **2FA** : TOTP (compatible Google Authenticator) avec activation par QR code et codes de secours à usage unique.
— **Verrouillage de compte** : Verrouillage progressif après des échecs répétés, avec temporisation exponentielle.

11.3. MODÈLE DE DONNÉES (AUTH_DB)

La table `users` stocke les comptes utilisateurs avec : email (unique), `password_hash` (Argon2id), indicateur d'activation TOTP, secret TOTP (lorsqu'activé), codes de secours (liste JSON hachée avec Argon2), horodatages, compteurs d'échecs et fenêtre de verrouillage. Les tables analytiques restent inchangées.

11.4. ACTIVATION ET UTILISATION DE LA 2FA

Lorsqu'un utilisateur active la 2FA, le serveur génère un secret Base32 aléatoire et affiche un QR code contenant une URI *otpauth* standard. L'utilisateur vérifie le premier code à 6 chiffres pour activer la 2FA. Le serveur génère 10 codes de secours à usage unique et n'en stocke que les versions hachées avec Argon2. À la connexion, si la 2FA est active, l'utilisateur doit fournir un TOTP valide ou un code de secours non utilisé.

11.5. SÉCURITÉ OPÉRATIONNELLE

— **Secrets** : `SECRET_KEY`, `AUTH_DATABASE_URI` et l'application de HTTPS sont fournis via des variables d'environnement dans Docker Compose.

— **Pas de mots de passe en clair** : Seuls des hachages Argon2id sont stockés ; les codes de secours sont également hachés.

— **Exposition minimale** : `auth_db` dispose de privilèges dédiés ; l'application utilise un compte au moindre privilège.

— **Résilience** : Le verrouillage et la limitation de débit réduisent l'impact des attaques par force brute et bourrage d'identifiants.

— **CSP et en-têtes** : Gérés par Talisman ; la CSP est initialement souple en raison de l'usage de CDN et peut être durcie.

11.6. INTERFACE UTILISATEUR

L'en-tête de chaque page inclut des boutons de connexion et d'inscription. Les utilisateurs authentifiés voient leur email et un bouton de déconnexion. L'activation de la 2FA propose l'inscription par QR code et le téléchargement des codes de secours.

11.7. SÉCURITÉ DES DONNÉES LORS DES MISES À JOUR

La chaîne d'import des données opère exclusivement sur `stops_db`. Le schéma `auth_db` est indépendant et n'est jamais supprimé ni réimporté, ce qui garantit la persistance des identifiants des utilisateurs lors des rafraîchissements de données.

11.8. PISTES DE DURCISSEMENT

— Imposer HTTPS dans tous les environnements et définir `SESSION_COOKIE_SECURE=true`.

— Ajouter la vérification d'email et la réinitialisation de mot de passe avec des jetons signés à durée limitée.

— Surveiller les en-têtes de sécurité et ajouter des listes d'autorisation CSP pour les ressources CDN ou auto-héberger les actifs statiques.

— Ajouter des journaux d'audit pour les actions d'administration et les événements d'authentification.

CHAPITRE 12 : EVALUATION DE LA SÉCURITÉ DE L'APPLICATION

CONCLUSION

Ce projet a cherché à approcher le problème de synchronisation des arrêts de transport public entre OSM et le système ATLAS en Suisse afin d'améliorer la précision des données de transport. Pour ce faire, nous avons mis en œuvre diverses méthodes de correspondance, telles que la correspondance exacte, par nom et par distance, tout en explorant la correspondance par itinéraire comme approche expérimentale. À ce jour, nous avons réussi à apparier 33 747 des 56 128 arrêts ATLAS. Par ailleurs, parmi les nœuds OSM sans correspondance actuelle, 27 759 sont associés à au moins un itinéraire, ce qui laisse entrevoir un potentiel significatif pour améliorer notre taux de correspondance grâce à l'intégration des données d'itinéraires.

Au-delà de ses aspects techniques, ce projet a constitué une expérience d'apprentissage particulièrement enrichissante. J'ai acquis des compétences en cartographie, dans les systèmes de routage des transports publics comme GTFS, ainsi qu'en développement web. Le défi de concevoir des algorithmes de correspondance et de rendre des données complexes accessibles m'a profondément stimulé, tout en mettant en lumière l'importance d'une communication claire dans les projets techniques. J'ai également trouvé fascinant d'explorer les cartes et de les examiner de près, car cela permet de mieux comprendre et apprécier le territoire suisse. Ce qui m'a le plus marqué, c'est le plaisir de découvrir ces domaines et de relever des défis qui, bien que complexes, se sont révélés passionnants.

En regardant vers l'avenir, plusieurs pistes d'amélioration se dessinent. On peut utiliser davantage d'informations disponibles, comme l'opérateur, parmi d'autres. Une analyse approfondie des balises et une importation prudente des nœuds OSM manquants s'imposent pour garantir l'intégrité des données. Un défi clé reste de déterminer, pour chaque correspondance, quel arrêt – OSM ou ATLAS – est le plus correct. Une solution envisagée serait d'améliorer notre application web et de l'ouvrir au public pour une vérification participative par des usagers familiers des zones concernées, ce qui nécessiterait d'importants efforts pour optimiser son ergonomie et ses fonctionnalités. Par ailleurs, la mise en place d'une structure de données robuste sera cruciale pour gérer et appliquer efficacement les corrections aux deux ensembles de données. Bien que le chemin à parcourir soit encore long, nos avancées actuelles renforcent notre détermination.

RÉFÉRENCES DOCUMENTAIRES

1. Open Data Platform Mobility Switzerland [en ligne]. Disponible sur : https://opentransportdata.swiss/en/ (consulté le 2024-12-05).
2. Traffic-points-actual-date [en ligne]. Disponible sur : https://data.opentransportdata.swiss/en/dataset/traffic-points-actual-date (consulté le 2024-12-12).
3. ATLAS App SBB [en ligne]. Disponible sur : https://atlas.app.sbb.ch (consulté le 2024-12-05).
4. Wikipédia. Liste des codes pays UIC [en ligne]. Disponible sur : https://fr.wikipedia.org/wiki/Liste_des_codes_pays_UIC (consulté le 2024-12-12).
5. OpenStreetMap Wiki. DE :Tag :highway=bus_stop [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/DE:Tag:highway=bus_stop (consulté le 2025-02-14).
6. OpenStreetMap Wiki. Public transport [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/Public_transport (consulté le 2025-02-14).
7. OpenStreetMap Wiki. FR :Key :public_transport [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/FR:Key:public_transport (consulté le 2025-02-15).
8. OpenStreetMap Wiki. Proposal :Public transport schema [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/Proposal:Public_transport_schema (consulté le 2025-02-15).
9. OpenStreetMap Wiki. Transport Map [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/Transport_Map (consulté le 2025-02-15).
10. OpenStreetMap Wiki. Key :local_ref [en ligne]. Disponible sur : https://wiki.openstreetmap.org/wiki/Key:local_ref (consulté le 2025-02-15).
11. Overpass Turbo [en ligne]. Disponible sur : https://overpass-turbo.eu/ (consulté le 2025-02-15).

Fin