Labeling vario-scale maps

Yan Gao
Student Number : 6006175

1st Supervisor: Martijn Meijers
2nd Supervisor: Peter van Oosterom

January 21, 2025

Contents

1 Introduction

Effective labeling is important for improving the readability and usability of maps. However, labeling vario-scale maps poses unique challenges. Traditional methods for placing labels work for fixed-scale maps, but vario-scale maps, where the scale changes continuously, require labels to adapt dynamically. For instance, as the map zooms in or out, labels must move or resize without overlapping with other labels or (dis)appearing abruptly. Poorly placed labels can make maps confusing and unexpected label movements can distract users. The problem is how to place and adjust labels so they stay readable, aligned with features, and that label transitions (such as resizing, repositioning, appearing, or disappearing) occur gradually and do not create sudden or jarring changes that disrupt user perception.

1.1 Problem statement and research goal

Many current map labeling methods were designed for fixed or limited-scale maps, so they do not handle the continuous scale changes in vario-scale maps very well. As users zoom in or out, labels can overlap, bump into each other, or appear and disappear suddenly, making the map confusing. Aligning labels with curved or long features (e.g., rivers and roads) adds extra difficulty. There is a lack of clear rules for determining which labels should remain visible when space is limited, prioritizing more important labels over less important ones. This creates a need for approaches that enable labels to adjust continuously in size and position, and appear or disappear gradually based on map scale changes, ensuring seamless transitions and enhancing readability and user experience.

This research aims to develop methods for effective label placement on vario-scale maps with which labels dynamically adjust in size, position, and visibility as the map scale changes (Figure 1). It will focus on finding optimal label positions that align with map features, ensuring smooth transitions without popping or sudden shifts. Additionally, the research will address overlapping labels by prioritizing more important ones and removing less critical labels to maintain clarity. The ultimate goal is to make vario-scale maps more readable and user-friendly while providing a seamless and visually smooth experience.

1.2 Scientific relevance

This research is important because well-placed labels could improve the usability of vario-scale maps. Although previous studies have explored static map labeling techniques, limited attention has been paid to the unique challenges posed by continuous scaling. Maps are increasingly used, and smooth transitions between scales are critical. By introducing potential methods for label placement, overlap management, and smooth transitions, this study may contribute to usage of vario-scale maps in practice.

2 Related Work

This section reviews existing researches on vario-scale mapping, map labeling methods, and font rendering techniques. The discussion begins with vario-scale techniques that enable smooth scale transitions, which may provide insights for integrating labeling methods that adapt to continuous zooming and panning. Next, labeling techniques for both static and dynamic maps are explored. Finally, the role of typography in map design is considered. This overview helps identify the main developments and gaps in the literature, providing a foundation for subsequent analysis.

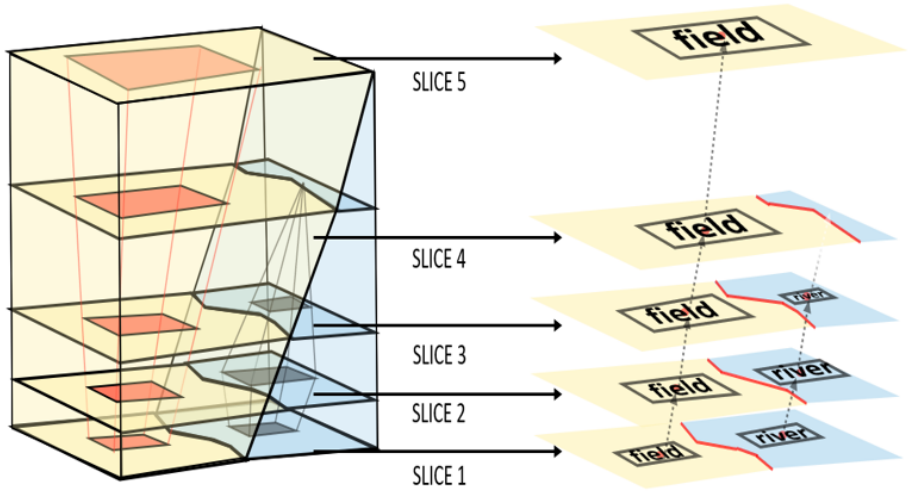


Figure 1: Example of vario-scale maps with labeling

2.1 Overview of vario-scale maps

Vario-scale maps are designed to provide seamless transitions between map scales, enabling continuous zooming and panning without the need for discrete representations. This approach contrasts with traditional multi-scale maps, which rely on predefined levels of detail and can result in abrupt changes in map content when zooming in or out. Vario-scale maps achieve a smooth transition through data structures and algorithms that integrate geometry, topology, and scale information.

2.1.1 Topological Generalized Area Partition (tGAP)

The Topological Generalized Area Partition (tGAP) is a vario-scale data structure designed to support map generalization across multiple levels of detail. It begins with a highly detailed planar partition, typically at the largest scale, and progressively simplifies the map by merging or collapsing less significant objects(van Oosterom and Meijers (2011)). These operations are recorded in a Directed Acyclic Graph (DAG), ensuring that all levels of detail are linked and accessible. The tGAP structure integrates both geometric and topological information, enabling efficient retrieval of representations at different scales. Figure 2 shows an example of a tGAP structure.

While the tGAP structure achieves a degree of scalability, its classic implementation introduces discrete transitions between scales, leading to abrupt changes in the map, such as objects suddenly disappearing or boundaries shifting, as shown in Figure 3. To address these limitations, the smooth tGAP was introduced as an extension of the classic model. The smooth tGAP incorporates a continuous scale dimension, enabling gradual transformations rather than abrupt ones. For example, objects shrink or grow progressively, and polygon boundaries adjust smoothly. These transitions are captured using tilted faces in a 3D space, creating a seamless representation of geographic features across scales. One example of smooth tGAP structure is as shown in Figure 4.

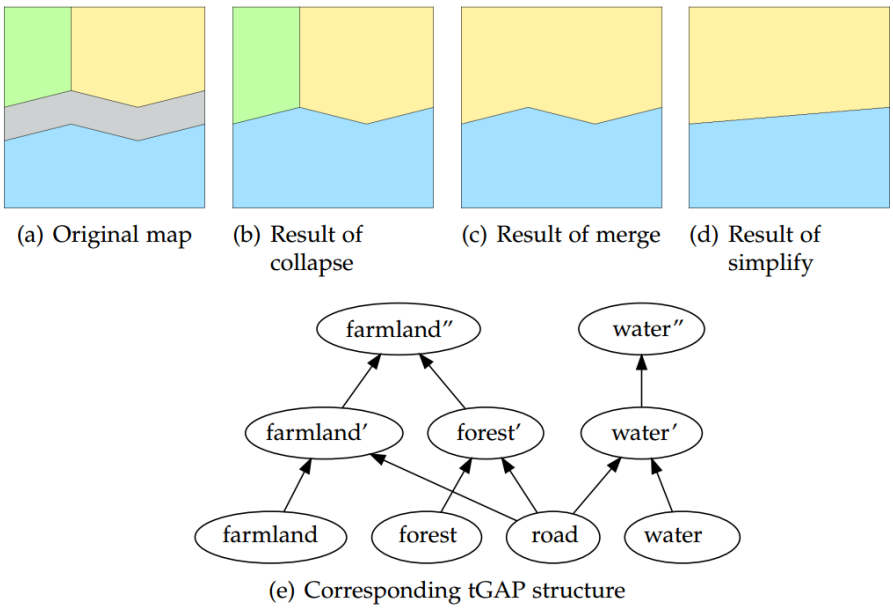


Figure 2: Map fragments and corresponding tGAP structure(van Oosterom and Meijers (2012b))

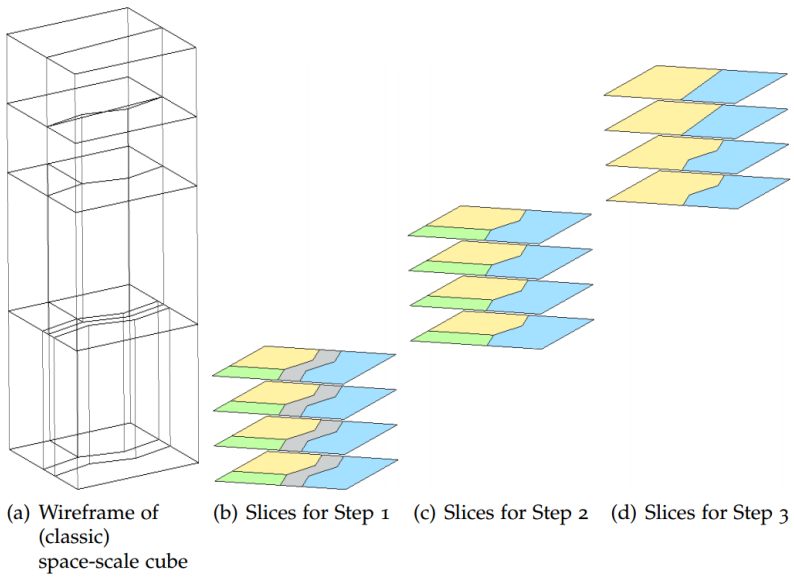


Figure 3: Map slices of the classic tGAP structure(van Oosterom and Meijers (2012b))

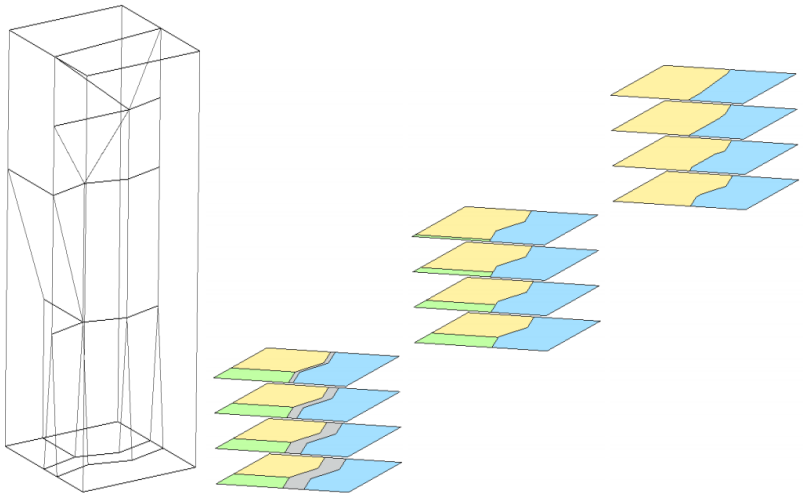


Figure 4: Map slices of the smooth tGAP structure(van Oosterom and Meijers (2012b))

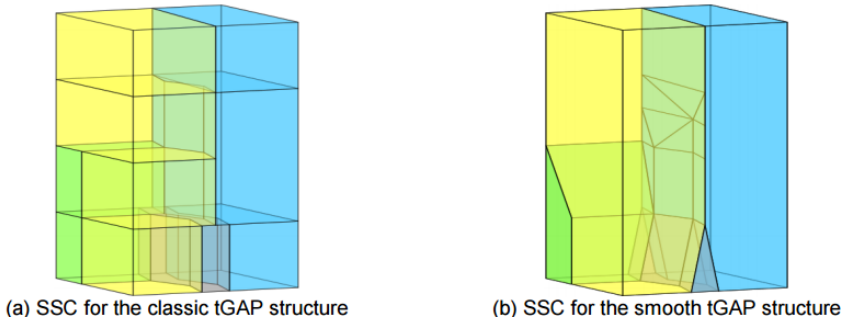


Figure 5: Space-scale cube(SSC) representation in 3D(van Oosterom and Meijers (2012b))

2.1.2 Space-scale cube (SSC)

The Space-Scale Cube (SSC) is a three-dimensional data structure designed to integrate spatial and scale dimensions, providing a foundation for true vario-scale maps(van Oosterom and Meijers (2012b)).

In the SSC, 2D features from a map are extruded into a third dimension, where the height represents scale. Polygons are represented as 3D prisms, lines as vertical faces, and points as vertical lines(van Oosterom and Meijers (2012a)). This approach provides a true vario-scale structure, where small changes in scale result in small, gradual changes in the geometry of features. Unlike the classic tGAP, which creates discrete snapshots at different scales, the SSC supports continuous representations. The comparison is shown in Figure 5.

Each 2D map can be derived by slicing the cube horizontally at a specific scale, yielding a planar partition, while non-horizontal or tilted slices can produce mixed-scale maps.

In this research, only horizontal slices are considered to derive 2D maps at specific scales. Non-horizontal or tilted slices, which produce mixed-scale maps, are excluded to simplify the implementation of label placement algorithms. Mixed-scale maps introduce varying levels of

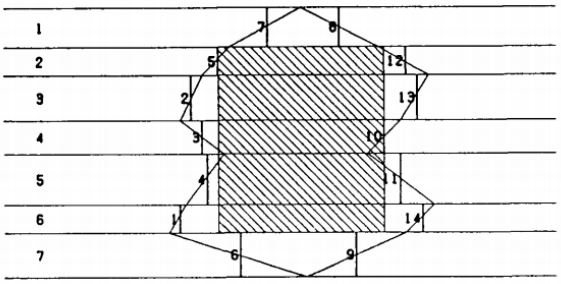


Figure 6: Simple polygon with strips, vertical line segments and sample box(van Roessel (1989))

detail within a single view, which can complicate label positioning as features from different scales may require different placement strategies within the same slice.

2.2 Label placement techniques

2.2.1 Static label placement

Static label placement techniques focus on optimizing label positions on fixed-scale maps, adhering to cartographic rules for clarity and spatial efficiency. While effective for static maps, these methods lack adaptability to dynamic scales, though their structured frameworks and optimization techniques may provide valuable insights for vario-scale labeling.

Edmondson et al. (Edmondson et al. (1996))proposed a general cartographic labeling algorithm that optimizes label placement for point, line, and area features on static maps. Their approach is structured into three key steps: candidate-position generation, position evaluation, and position selection. By using simulated annealing to minimize overlaps and maximize clarity, the algorithm produces visually appealing labels for dense, page-sized maps.

Van Roessel (van Roessel (1989)) introduced an algorithm for locating candidate labeling boxes within polygons, addressing challenges where centroid-based label placement results in awkward or overlapping positions. The method divides a polygon into horizontal strips, generates vertical segments within these strips, and identifies maximal candidate boxes that are wholly contained within the polygon(Figure6), allowing flexible label placement and repeating labels for large polygons.

Barrault (Barrault (2001)) also introduced a methodology specifically for labeling area features, addressing challenges such as complex shapes and overlapping constraints. The process incorporates morphological operations(erosions)(Figure7) to simplify area boundaries, skeletonization to extract important geometric features(Figure8), and a quality function to evaluate label placements based on their perceived coverage and conformity to the area shape.

Similarly, Dörschlag et al. (Dörschlag et al. (2003)) proposed an algorithm for placing objects, such as labels or diagrams, within map areas while maintaining cartographic quality. The approach involves a two-step process: erosion of the area boundary to identify feasible placement regions for objects and reduction of these regions to optimal positions using the skeleton of the eroded area(Figure9). The scoring function evaluates candidate positions based on criteria like proximity to the area centroid and skeleton nodes with high connectivity.

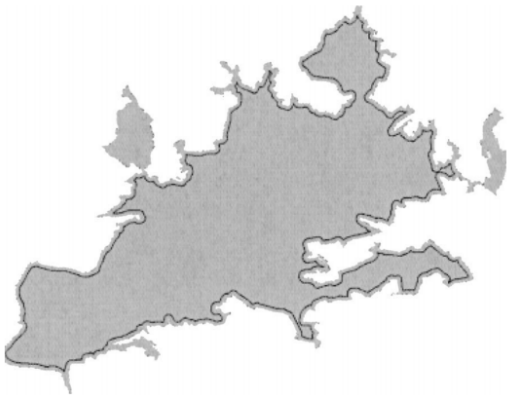


Figure 7: Morphological erosion to extract the most important part(Barrault (2001))

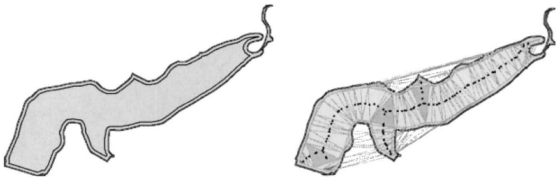


Figure 8: Eroded area and its skeleton(Barrault (2001))

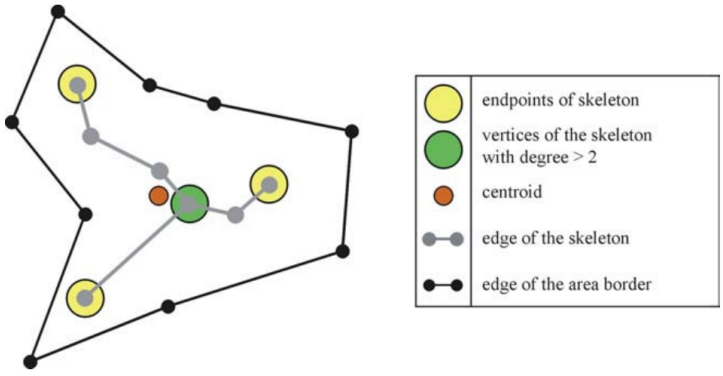


Figure 9: Special points of the area and the skeleton(Dörschlag et al. (2003))

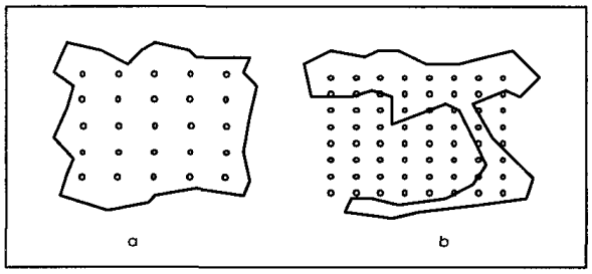


Figure 10: Anchor points as candidates for the start and end points of the text's baseline(Pinto and Freeman (2006))

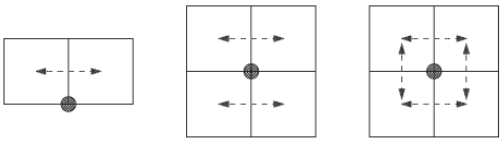


Figure 11: Top-, two- and four-slider model(van Kreveld et al. (1999))

Pinto and Freeman (Pinto and Freeman (2006)) introduced a feedback-based method for the automated placement of text labels within bounded regions on maps, emphasizing conformity to region size and shape. The method includes a two-step feedback process: an initial placement is generated based on anchor points within the region(Figure10), and the placement is iteratively refined by evaluating it against quality criteria such as length, symmetry, clearance, and conformity. This iterative refinement allows the system to handle the complex and varied geometries more effectively than static algorithms.

Wolff et al. (Wolff et al. (2001)) proposed a simple and efficient algorithm for labeling linear features, such as rivers or roads, on maps. Their method creates a candidate strip along the polyline, ensuring that labels follow cartographer Imhof's ealier work(Imhof (1975)) for line labeling by satisfying hard constraints (minimum distance, curvature limits, and non-intersection) and optimizing soft constraints (proximity to the polyline, minimal inflections, and horizontal alignment).

Barrault and Lecordix(Barrault and Lecordix (1995)) developed an automated system for linear feature name placement that adheres to stringent cartographic quality criteria. Their methodology involves dividing roads into sections for labeling, generating candidate positions along straight parts of the road, and evaluating these positions based on intrinsic quality measures such as curvature, local centering, and background overlap. Simulated annealing is used to optimize label placement locally, ensuring regular spacing along roads while avoiding conflicts with other map features. Its focus on maintaining spatial harmony and addressing label repetition offers valuable insight.

Van Kreveld et al. (van Kreveld et al. (1999)) proposed algorithms for point labeling using sliding label models(Figure11), which allow labels to continuously move along the edges of a rectangle instead of being restricted to a finite set of positions.

Strijk and van Kreveld (Strijk and Kreveld (2000)) extended the sliding label model for point labeling by introducing practical constraints such as obstacles and varying label heights. Their algorithm efficiently ensures that labels avoid overlapping with obstacles, such as boundaries,

while maintaining proximity to the labeled point.

Static label placement techniques offer structured frameworks for optimizing label positions, minimizing overlaps, and ensuring cartographic clarity. Methods such as skeletonization, candidate position generation, and simulated annealing provide solid solutions for handling dense maps, complex geometries, and specific feature types (e.g., lines, areas, and points). However, these methods also have notable limitations. They are inherently designed for static, fixed-scale contexts and do not adapt dynamically to user interactions like zooming. In summary, while static label placement methods offer insights into optimal label placemens, their inability to handle continuous scale transitions and user interactions highlights the need for new approaches that support vario-scale maps.

2.2.2 Dynamic label placement

Dynamic label placement techniques focus on the challenges of real-time labeling in interactive maps, ensuring smooth transitions during zooming, panning, and scaling. They highlight the importance of balancing computational efficiency with cartographic quality, providing insights for adapting dynamic principles to vario-scale map labeling systems.

Been et al. (Been et al. (2006)) introduced a framework for dynamic map labeling, addressing challenges of continuous zooming and panning while maintaining label consistency and interactive performance. They proposed desiderata for dynamic label consistency:

- Monotonicity of labels: labels should not appear, disappear, and reappear unpredictably as the user zooms in or out.
- Continuous transitions: label positions and sizes should change smoothly during panning and zooming.
- Pan consistency: labels should not unexpectedly vanish or appear during horizontal or vertical panning.
- History independence: the placement and selection of labels should depend solely on the current map state (scale and view), not the interaction path taken to reach it.

To address above challenges, the authors proposed an invariant point placement approach, where each label is associated with a fixed point on the map, ensuring that it stays in place relative to the feature being labeled. Labels maintain a constant screen size across scales, ensuring readability without abrupt shifts. This placement method leads to cone-shaped ”extrusions” in a space-scale representation, where the label's visibility at different scales forms a continuous 3D cone, as shown in Figure12. The idea of invariant point placements and active ranges offers a possible starting point for exploring how labels can appear and disappear in a user-friendly manner during continuous scale transitions.

The approach prioritizes computational efficiency and guarantees consistent label visibility during interactions. However, the method restricts label positions to avoid ”sliding” along features, which the authors consider an acceptable trade-off for performance and stability.

Building upon the earlier work, Been et al. (Been et al. (2010)) proposed the Active Range Optimization (ARO) framework for consistent dynamic map labeling. The ARO problem focuses on selecting the optimal scale intervals during which labels remain visible (i.e., their ”active ranges”) while maximizing label presence across scales and avoiding conflicts. Their approach

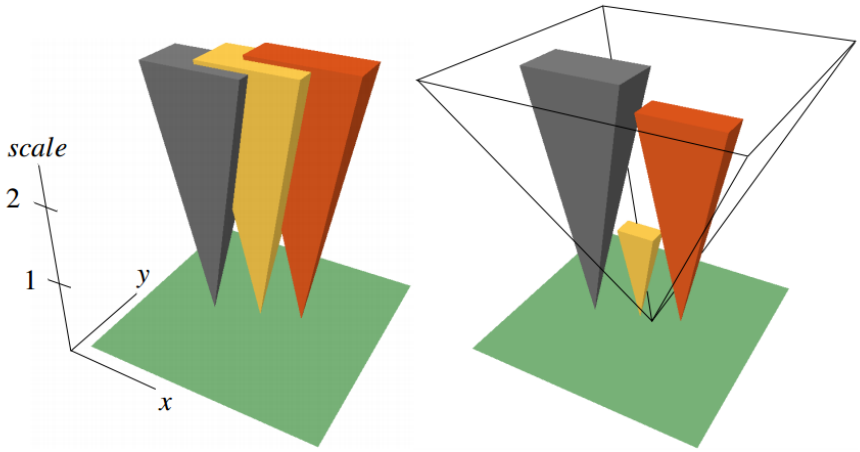


Figure 12: Dynamic placements for three labels(without and with active ranges)(Been et al. (2006))

involves considering labels as geometric shapes (cones or rectangles) extruded across the scale dimension and proposes methods to approximate optimal active ranges using dynamic programming, greedy algorithms, and divide-and-conquer techniques.

Schwartges (Schwartges (2015)) explored dynamic label placement for interactive maps, emphasizing real-time adaptability during user interactions like zooming and panning. In Section 3.3, the focus is on two greedy heuristics: shrinking-cones and growing-cones. The shrinking-cones method starts with full visibility for labels and iteratively reduces their active ranges to resolve conflicts, whereas the growing-cones method begins with minimal active ranges and gradually expands them until conflicts arise, and when it happens, the system removes labels in crowded areas to make the map easier to read(Figure13). Both heuristics aim for fast computation and near-optimal solutions, balancing performance with cartographic quality.

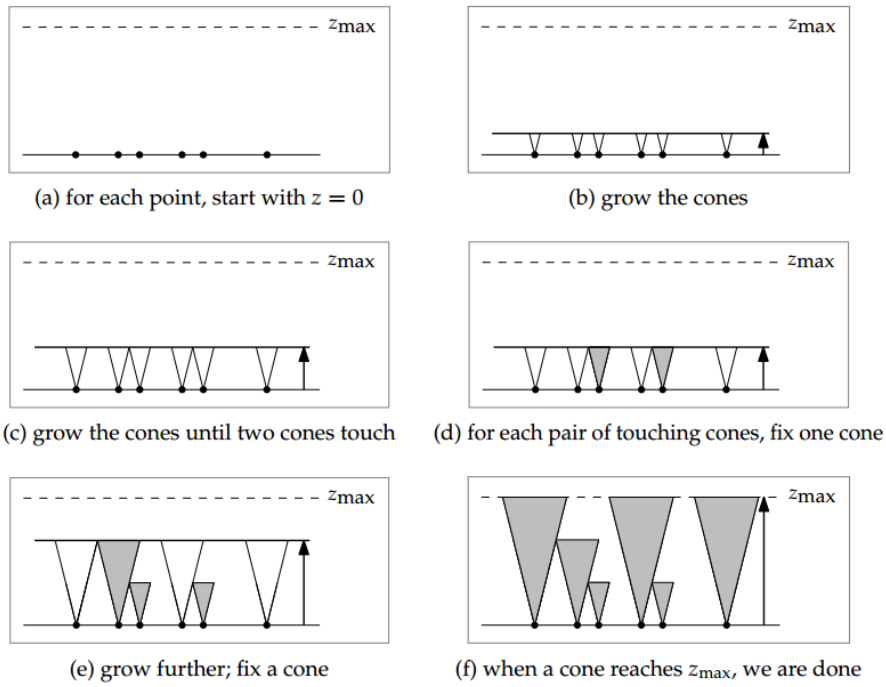
Both heuristics aim to maximize label presence over varying scales while maintaining computational efficiency, as exact solutions are computationally expensive for real-time applications. The experiments in this section also highlight that while greedy methods may not always yield optimal results, they produce near-optimal solutions with minimal computational overhead.

2.2.3 Existing labeling library

The PAL (Placement Automatique des Labels) library(Ertz et al. (2008)), addresses the complex challenge of automated cartographic label placement. The project creates an efficient tool for label placement on maps. The library's primary goal is to maximize the number of displayed labels while avoiding overlaps, thereby enhancing map readability and usability.

PAL supports multi-layer labeling for points, lines, and polygons, with customizable options such as:

- Layer prioritization: determines which labels appear when conflicts arise.
- Obstacle detection: prevents labels from overlapping with map features.



(a) for each point, start with $z = 0$ (b) grow the cones

(c) grow the cones until two cones touch (d) for each pair of touching cones, fix one cone

(e) grow further; fix a cone (f) when a cone reaches z_{max}, we are done

Figure 13: Illustration of growing-cones heuristic(Schwartges (2015))

- Orientation preferences: supports various label placements (e.g., horizontal, centroid-based, line-following).

The library handles labels as abstract bounding boxes, and operates in two primary phases: (1) problem generation to identify candidate label positions and constructs a conflict graph indicating which labels cannot co-exist; (2) optimization to choose optimal label placements using combinatorial optimization techniques.

2.3 Typography in map labeling

Typography is an important yet often underutilized aspect of cartographic design, playing a vital role in conveying geographic information and enhancing map readability. Additionally, efficient font rendering is important for vario-scale map labeling, where labels must dynamically adapt to changes in scales.

2.3.1 Font selection in cartography

Biniek et al. (Biniek et al. (2019)) highlight how typographic choices can visually differentiate toponymic categories, such as settlements, hydrography, and transport networks, through variations in font, size, color, and style. For instance, Google Maps uses the Roboto font with size variations and uppercase text to emphasize hierarchy, while OpenStreetMap (OSM) uses Noto Sans, relying on italics and colorimetric differences for category distinction. In Figure14, variations in size and uppercase text are used to emphasize the hierarchy in cities, city regions and landmarks. In Figure 15, highway labels are shown in bold and red for emphasis, while park names are styled in italics with a greenish hue to align with their background.

Typography, including font type, size, style (e.g., italics), and color, can visually encode the importance and function of labels. This encoding could be useful in vario-scale maps to ensure labels adapt effectively as the map scale changes.

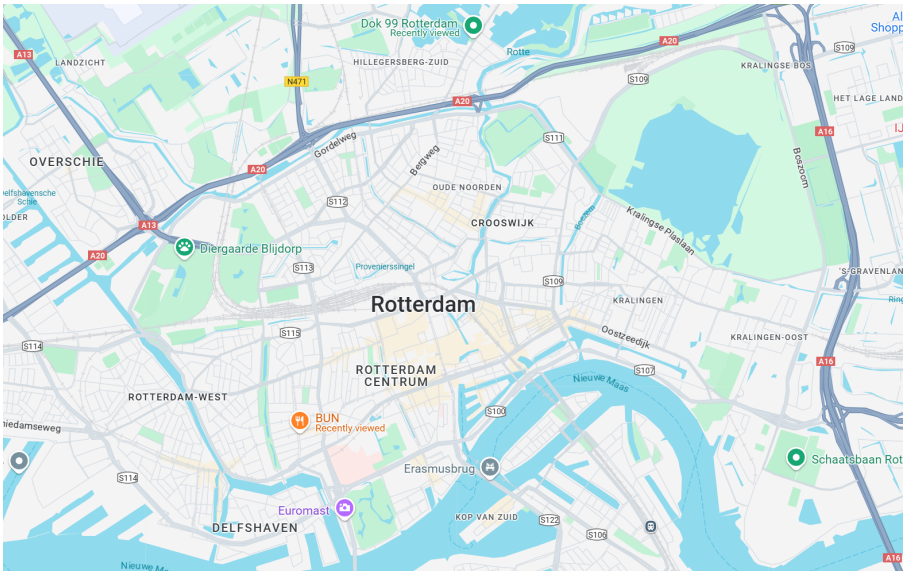


Figure 14: Rotterdam area in Google maps: font sizes and uppercase show hierarchy

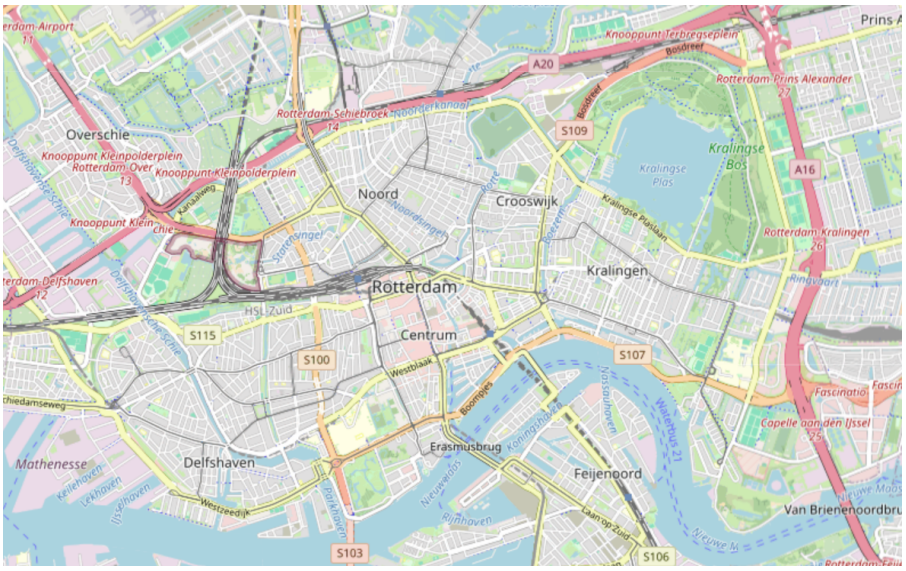


Figure 15: Rotterdam area in OSM maps: italics and colorimetric differences show hierarchy

2.3.2 Font rendering techniques

Font rendering is a key consideration in vario-scale maps labeling to ensure that text remains sharp and readable as the map scale changes. One effective technique is using Signed Distance Fields (SDFs)(Green (2007)), which store information about the shape of a font in a compact and efficient way. This allows text to be rendered smoothly at any zoom level, avoiding common problems like blurriness or jagged edges seen with traditional methods. SDFs also make it possible to apply dynamic styles such as outlines, shadows, or soft edges without needing large font files, ensuring labels remain visually appealing and readable. Importantly, this technique works well across different devices, making it suitable for interactive and web-based map applications. By integrating this font rendering techniques, vario-scale maps may achieve clear and consistent label appearance across scales.

3 Research Questions

3.1 Main question and sub-questions

The main research question of this thesis is:

How can labels be dynamically placed and adjusted on vario-scale maps to maintain readability, usability, and visual coherence across continuously changing scales?

Following sub-questions break down this broad question into specific components that address various technical and design challenges:

- 1. **Label selection and prioritization**: What criteria should be used to determine which labels (e.g., house numbers, road/river names, neighborhoods) should appear or disappear at different scales? How can label importance be quantified?
- 2. **Optimal placement techniques**: How can the optimal positions for labels be determined for both regular and irregular features, ensuring spatial alignment and visual clarity? Specifically, what criteria (e.g., minimizing overlap between labels, maintaining proximity to the labeled feature, and preserving label readability) should be used to define optimal placement?
- 3. **Dynamic adjustments**: How can dynamic resizing and repositioning of labels be managed to avoid overlaps with other labels while maintaining readability during continuous scale transitions?
- 4. **Font selection and rendering**: What font attributes (e.g., typeface, size, style) and rendering techniques can ensure readability and visual consistency of labels across different scales?
- 5. **Data structure and retrieval**: How can label information be efficiently structured, stored, and transmitted between the server and client to support rendering of vario-scale maps with labels?

3.2 Scope

This research aims to address the specific challenges associated with dynamic labeling on vario-scale maps, focusing on adapting labels to continuously changing scales.

The research will focus on exploring practical approaches for label placement, including positioning, resizing, and transitioning, to maintain clear and coherent map readability across scales. This may involve existing tools and frameworks where appropriate, with necessary adaptations to support vario-scale maps. The study will include prototyping and testing on existing vario-scale maps to evaluate the effectiveness of different approaches and try to achieve smooth label transitions during scale changes.

This research excludes certain areas to maintain a focused approach. It excludes map rotation as a factor. All methods and evaluations will assume a fixed map orientation to simplify the scope and focus on other core challenges. It will not address labeling for multi-language or non-Latin scripts, limiting its scope to English characters. Additionally, the study is confined to 2D vario-scale maps and does not extend to 3D mapping.

In this research, pre-existing label data will be used, which refers to datasets where the labels (e.g., place names, street names, and points of interest) are already associated with their corresponding geometries in the map. This research does not focus on creating new label content (e.g., automatically generating street names or missing labels) but rather on optimizing the placement and rendering of existing labels across scales. However, pre-processing steps, such as extracting labels from spatial datasets, will be included as part of the methodology.

Furthermore, the research does not aim to develop new methods for text rendering but rather integrate existing techniques to achieve dynamic and visually coherent label placement.

4 Methodology

4.1 Label prioritization

This section outlines the method for determining an importance value for each label in vario-scale maps. The importance value will quantify the priority of the appearance of a label.

First establish tiers to reflect the relative importance of different label types. Labels in a higher tier will always take precedence over labels in a lower tier(labels in Tier *n* always take precedence over labels in Tier *n+1*).

- Tier 1: critical features (e.g., major cities, capitals, large bodies of water).
- Tier 2: important features (e.g., towns, primary highways, major rivers).
- Tier 3: ordinary features (e.g., villages, secondary roads, small parks).
- Tier 4: minor features (e.g., local roads, minor landmarks, small rivers).

For labels representing populated places (e.g., cities, towns, villages), calculate scores based on population. Use ranges to assign normalized scores, for example(as in Table 1):
If two place labels conflict, the one with the higher population score takes precedence.

Assign scores to feature types based on their relative importance within the same tier, for example, here are scores for Tier 3 features(see Table2):

Combining population data with feature types, a composite score could be calculated:

$$S = T + (w_1 \cdot P + w_2 \cdot F)$$

Population Range	Normalized Score
$> 1,000,000$	1.0
500,000–1,000,000	0.8
100,000–500,000	0.6
10,000–100,000	0.4
$< 10,000$	0.2

Table 1: An example of population range and normalized scores for label prioritization

Feature Type	Normalized Score
Secondary roads	0.6
Small rivers	0.5
Villages	0.4
Local parks	0.3

Table 2: Feature type and normalized scores for label prioritization

Where:

- T: Tier multiplier, a large constant that ensures higher tiers always precede lower tiers.
  - Example: $T_1 = 1000, T_2 = 100, T_3 = 10, T_4 = 1$.
- P: Population score (normalized within the tier).
- F: Feature type score (normalized within the tier).
- w_1, w_2: Weights for population and feature type (e.g., $w_1 = 0.7, w_2 = 0.3$).

4.2 Label placement

One possible way to label vario-scale maps could be to use key slices of vario-scale maps and calculate labels for each slice while ensuring smooth transitions between them. Since tGAP organizes the data hierarchically across scales, it can be used to query and retrieve features relevant for each key slice. This approach needs to combine discrete label computations at key scales (using PAL) with interpolation techniques to create continuous and seamless label transitions.

1. **Query geometries for a key scale**: As mentioned in section2.1.1, to calculate labels for a specific key frame, the tGAP structure could be queried to determine the features that exist at the chosen scale. The tGAP organizes map features hierarchically across scales, meaning it can return the relevant geometries (points, lines, or polygons) that correspond to the desired zoom level. This prepares the map data for input into the PAL labeling process.

2. **Label placement with PAL for the key slice**: Once the relevant geometries are retrieved from the tGAP for the current slice, these features could be passed as input layers to PAL for label placement. PAL generates candidate label positions for each type of feature - points, lines, and polygons - based on configured settings. PAL then optimizes the placement to resolve overlaps and ensure that the most relevant labels are displayed. The result of this process is a set of optimized, conflict-free label positions that are saved for rendering in the key frame.

3. **Smooth transitions between key slices**: Labels should move smoothly as features morph, split, or merge across scales.

- Position transitions: The position of a label is interpolated linearly as the user zooms or pans between key slices. As the zoom level changes, the label's coordinates shift proportionally to reflect the updated geometries. For example, as the map transitions from a regional to a city-level view, city labels may move smoothly to more precise positions on the map. Linear interpolation could ensure that the labels stay aligned with their corresponding features, avoiding sudden jumps.
- Opacity transitions: Opacity transitions are used to handle labels that appear or disappear at certain zoom levels. For labels that exist in both key slices, opacity remains fully visible throughout the transition. However, for labels that only appear at more detailed slices, the opacity is interpolated from transparent to opaque as the user zooms in (or vice versa when zooming out). This fade-in/fade-out effect prevents labels from popping in or out abruptly.
- Snapping mechanism for opacity: To avoid awkward half-transparent labels when the user pauses mid-zoom, a snapping mechanism could be applied. If the user stops between two key slices, the opacity shall 'snap' to either fully visible or fully hidden over a short transition period (e.g., 200 milliseconds). This brief fade ensures that labels do not remain in an intermediate, visually ambiguous state.

Another possible way is to derive optimal achor points from the most detailed level of a vario-scale map, and then in combination with Active Range Optimization (ARO) and growing cones as mentioned in section2.2.2 to assign lifetimes to labels.

1. **Optimal candidate position generation at the most detailed level**: Using the PAL labeling library, the optimal candidate positions for each feature type (points, lines, polygons) are generated based on spatial preferences. Once the optimal positions are determined, they are saved as anchor points for label placement across all scales.

2.**Define growing cones for each anchor point**: Each anchor point is assigned a growing visibility cone that defines the range of scales during which the label is visible.

3.**Active Range Optimization (ARO) for label lifetimes**: ARO detects when cones for different labels intersect, indicating potential overlaps. When overlaps occur, ARO adjusts the lifetimes of lower-priority labels by shrinking their visibility cones, ensuring that higher-priority labels remain visible. High-tier labels (e.g., city names) retain their full cone, while lower-tier labels (e.g., street names) may be limited in their range.

4.**Labels during scale transitions**: Labels gradually fade in at the start of their lifetime and fade out at the end, preventing abrupt appearances or disappearances. Since the same anchor points are used across scales, label positions remain consistent, moving smoothly as the user zooms, while the label size may dynamically adjust (e.g., from a larger font at coarse scales to a smaller one at detailed scales) while retaining readability.

4.3 Font rendering

After the labels are placed and optimized, they are rendered with smooth animations for position changes, opacity transitions, and size adjustments.

Different types of features on the map (e.g., roads, rivers, administrative boundaries, and points of interest) may apply specific fonts to enhance their visual distinctiveness. Some considerations may guide font selection:

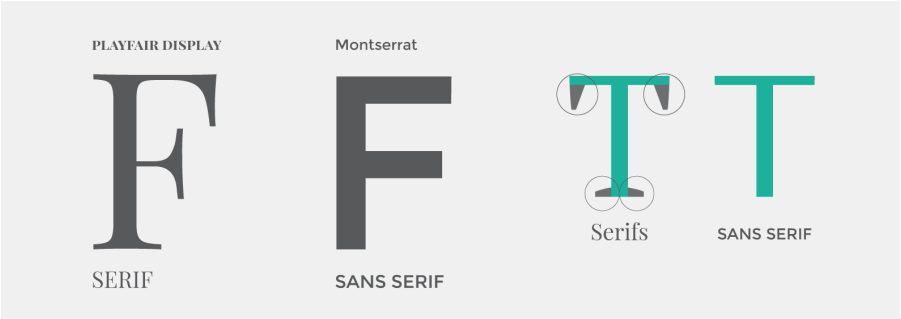


Figure 16: Difference between Serifs and Sans-Serif

- Font styles: For example, Serif fonts may be used for formal elements like large cities, while Sans-Serif fonts are preferred for roads and urban areas due to their clarity at small sizes(Figure16).
- Font boldness: Key features, such as major highways or important rivers, may use bolder fonts to emphasize their importance.

To render label texts, SDF-based method is a resolution-independent way for text rendering by encoding the distance of each point in the glyph to its nearest edge, which enables smooth text scaling and special effects such as outlines and shadows:

Before rendering labels, the font must be converted into a font atlas - an image containing SDF representations of the glyphs and metadata describing their layout and spacing. The texture atlas containing all glyphs and a metadata file describing the glyph positions, sizes, and kerning pairs shall be obtained.

Once the SDF font is preprocessed, text can be added as text meshes. Text objects shall be rendered by referencing the preprocessed SDF font atlas and applying shaders to control how glyphs are drawn.

To enhance visual quality and ensure readability, an additional overlap detection mechanism shall be applied during the rendering process. This step evaluates overlaps between the rendered bounding boxes of labels, taking into account variations in font style, size, and effects.

If overlaps are detected during rendering:

- Lower-priority labels may fade out gradually.
- Font sizes may be adjusted within permissible limits to fit constrained spaces.

The real-time overlap detection complements the previous placement step, ensuring that dynamic user interactions, such as zooming or panning, do not compromise the map's readability.

4.4 Label data structure and retrieval

Efficiently structuring, storing, and transmitting label information is important for real-time rendering of vario-scale maps.

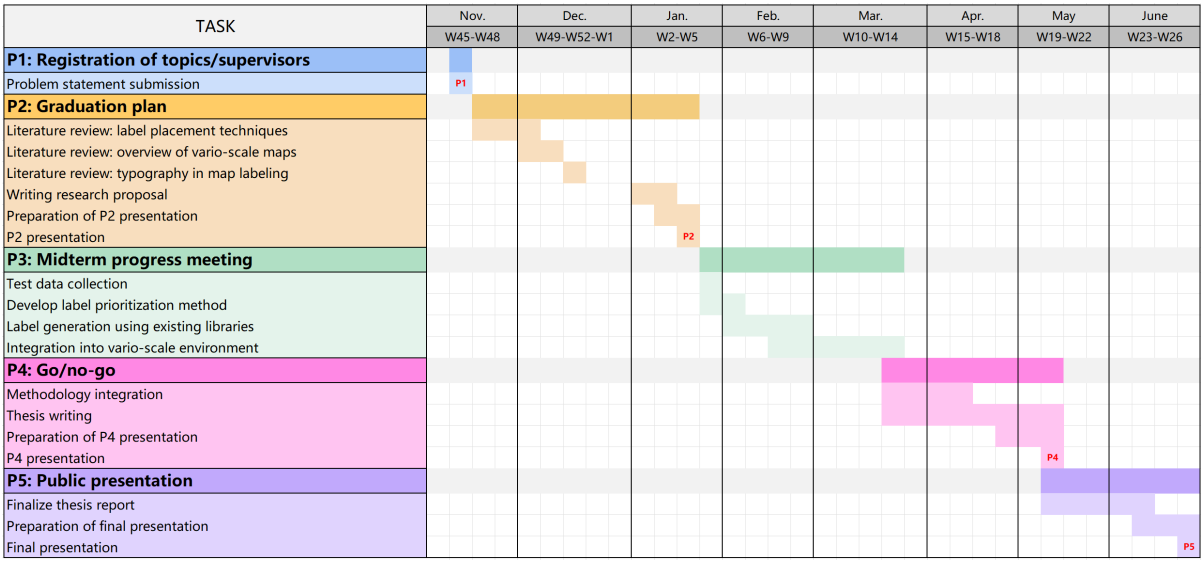


Figure 17: Schedule

Each label entry includes metadata such as a unique identifier, anchor point, precomputed importance score, and visibility range specifying the minimum and maximum zoom levels. Additionally, font attributes, such as type, size, and style, are also included. The label data could stored in a spatial database such as PostgreSQL/PostGIS, which supports spatial queries and indexing for fast access to geometries and corresponding labels.

To minimize data transmission between the server and client, only the labels relevant to the user's current viewport are transmitted. The server filters label data based on the requested zoom level and spatial bounds before sending it to the client. A client-side caching mechanism could store previously loaded labels to avoid redundant data requests.

The overall data retrieval and rendering process involves several steps. When the user zooms or pans, the client sends a request to the server for the labels relevant to the new viewport. The server responds with filtered, precomputed label data, with which the client then parses and renders using a rendering framework such as WebGL. The font glyphs referenced by the labels are loaded from the preprocessed SDF font atlas to ensure sharp and scalable text. The client dynamically updates the viewport while reusing cached data where possible to reduce unnecessary requests.

5 Schedule

The research schedule is shown in Figure17, where important Ps are highlighted. Regular meetings with supervisors will be arranged once every two weeks.

6 Tools and Datasets

6.1 Tools

1. Database management systems(e.g. PostgreSQL with PostGIS)
A database management system could be used for organizing and managing spatial data. PostgreSQL with PostGIS may support storage, querying, and analysis of geographic and

spatial data.

2. Existing libraries and toolkits
Using existing libraries can enhance the development process by providing optimized algorithms and rendering capabilities.

- WebGL: Used for rendering vario-scale maps directly in the browser, enabling display of geometry and labels.
- tGAP(https://github.com/bmmeijers/tgap-ng): A code repository that supports the generation of the tGAP structure for vario-scale map representations.
- PAL (http://pal.heig-vd.ch/): A cartographic labeling library designed for defining optimal label placements.
- Font rendering (e.g., PIXI.js BitmapFont): A JavaScript-based tool for creating bitmap fonts, which can be helpful for fast text rendering on web-based maps. [Documentation: https://pixijs.download/release/docs/text.BitmapFontData.html]
- TU Delft Vario-scale maps demo(https://varioscale.bk.tudelft.nl/gpudemo/2022/01/top10nl/): The demo itself provides the rendering environment and vario-scale structure, which could be learned from.
- Performance profiling tools(e.g. Chrome DevTools): Browser-based profiling tools, such as Chrome DevTools, may be used to monitor performance during zooming and panning interactions, track memory usage, and measure rendering times and frames per second (FPS).

3. Programming tools
Programming environments help working with libraries, creating custom algorithms, and debugging.

- Visual Studio (C++): An IDE for developing and integrating C++ libraries.
- PyCharm (Python): Useful for working with Python-based libraries.
- Visual Studio Code(HTML; CSS; JavaScript): Useful for web application development.

4. GIS software (e.g., QGIS)
GIS software is useful for visualizing and validating intermediate results. QGIS is an open-source GIS software with visualization abilities.

6.2 Datasets

TOP10NL dataset(https://www.pdok.nl/introductie/-/article/basisregistratie-topografie-brt-topnl)
For this research, the TOP10NL dataset, a comprehensive Dutch topographic dataset containing various geographic feature layers could be used. The dataset provides detailed spatial geometries (points, lines, and polygons) along with corresponding names for geographic features, making it suitable for the purposes of labeling vario-scale maps.

Selected feature layers
A tGAP structure will be included in this research, thus the selected layers from exsiting

datasets(e.g., built-up areas, geographical regions) should be polygonal and form a consistent partition without overlaps or gaps.

Research area
The Utrecht region is an ideal test area due to its dense and diverse set of features, such as buildings, roads, and waterways. Additionally, its combination of urban areas and surrounding rural regions makes it ideal for testing labels with varying priorities and densities.

Population dataset(https://www.cbs.nl/nl-nl/dossier/nederland-regionaal/geografische-data/gegevens-per-postcode)
The dataset provides population statistics at the postal code level, which can be aggregated to calculate populations for cities, towns, and villages.

7 Research Plan and Assessment

This section provides details about the steps to be conducted in this research, including data selection, geographic focus, development phases, and assessment criteria for evaluating the outcomes.

7.1 Data selection and geographic focus

- Data source: The primary dataset used will be the TOP10NL dataset, which includes geographic features relevant to the Netherlands. The dataset includes feature layers such as buildings (gebouw_vlak), functional areas (functioneel_gebied_vlak), geographic regions (geografisch_gebied_vlak), and road sections (wegdeel_vlak).
- Geographic focus: The Utrecht region is selected due to its diversity in urban, suburban, and rural features, providing a test environment for evaluating label prioritization and placement.

7.2 Research steps

1. Preprocessing and database setup:
The first step involves importing the selected TOP10NL layers into a spatial database (PostgreSQL with PostGIS) to manage and query the spatial data. The data is clipped to the administrative boundary of Utrecht using geospatial queries. Label information (e.g., building names and street names) is associated with the corresponding geometries for following label placements.

2. Label prioritization:
To determine the importance of each label, a scoring system is established(as described in section4.1). Population data is used to prioritize populated place labels, while feature type scores (e.g., highway vs. park) are applied to other labels. A composite importance score is calculated to handle conflicts between overlapping labels.

3. Label placement:

- Key slices with interpolation: In this method, labels are computed at specific zoom levels (key slices), and their positions are interpolated between these levels to create smooth transitions. Labels are dynamically merged or split when generalized features change across scales.

- Anchor points with Active Range Optimization (ARO): This method uses precomputed optimal anchor points for each feature at the most detailed level. Each label is assigned a visibility range defined by a ”growing cone” structure, indicating its active scale range. ARO manages overlaps by adjusting the visibility of lower-priority labels.

4. Label data structure and retrieval:
The label information is structured in a hierarchical format, organized by spatial partitions(e.g. quadtree). Label metadata includes geometry, importance scores, and visibility ranges, which shall be stored at a spatial database with spatial indexing. The client fetches only the relevant labels based on the viewport and caches the data to minimize redundant requests.

5. Prototype development:
A prototype interactive map shall be developed using WebGL. This prototype shall integrate precomputed label metadata, the tGAP structure, SSC, and SDF font atlases. The implementation focuses on rendering labels smoothly across various zooming and panning.

6. Testing and validation:

- Rendering performance: Measure the average response time during zooming and panning. Performance could be tracked using browser-based profiling tools (e.g., Chrome DevTools) to record the time taken for each interaction and the frames per second (FPS).
- Memory usage: Track the peak memory consumption during interactions to detect memory inefficiencies.
- Visual quality: Validate that labels remain aligned with their corresponding geometries, avoid overlap with other labels, and transition smoothly.

7.3 Assessment criteria

- Performance: Labels should render smoothly, with a relatively low average response times and low memory usage.
- Visual quality: Labels should transition smoothly and avoid abrupt changes.

References

M. Barrault. A methodology for placement and evaluation of area map labels. *Computers, Environment and Urban Systems*, 25(1):33–52, 2001. doi: 10.1016/s0198-9715(00)00039-9.

M. Barrault and F. Lecordix. An automated system for linear feature name placement which complies with cartographic quality criteria. In *AUTOCARTO-CONFERENCE-*, pages 321–330, 1995.

K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006. doi: 10.1109/tvcg.2006.136.

K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry*, 43(3):312–328, 2010. ISSN 0925-7721. doi: https://doi.org/10.1016/j.comgeo.2009.03.006. URL `https://www.sciencedirect.com/science/article/pii/S0925772109000649`. Special Issue on 24th Annual Symposium on Computational Geometry (SoCG'08).

S. Biniek, G. Touya, and G. Rouffineau. Fifty shades of roboto: Text design choices and categories in multi-scale maps. *Advances in Cartography and GIScience of the ICA*, 1:1–8, 2019. doi: 10.5194/ica-adv-1-2-2019.

D. Dörschlag, I. Petzold, and L. Plümer. Placing objects automatically in areas of maps. 01 2003.

S. Edmondson, J. Christensen, J. Marks, and S. Shieber. A general cartographic labelling algorithm. *Cartographica*, 33(4):13–24, 1996. doi: 10.3138/U3N2-6363-130N-H870. URL `https://doi.org/10.3138/U3N2-6363-130N-H870`.

O. Ertz, M. Laurent, D. Rappo, A. Sae-Tang, and E. Taillard. Pal-a cartographic labelling library. 01 2008.

C. Green. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, page 9–18, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781450318235. doi: 10.1145/1281500.1281665. URL `https://doi-org.tudelft.idm.oclc.org/10.1145/1281500.1281665`.

E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975. doi: 10.1559/152304075784313304.

I. Pinto and H. Freeman. *The feedback approach to cartographic areal text placement*, volume 1121, pages 341–350. 01 2006. ISBN 978-3-540-61577-4. doi: 10.1007/3-540-61577-6_35.

N. Schwartges. *Dynamic Label Placement in Practice*. doctoralthesis, Universität Würzburg, 2015.

T. Strijk and M. Kreveld. Practical extensions of point labeling in the slider model. *GeoInformatica*, 6, 03 2000. doi: 10.1023/A:1015202410664.

M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry*, 13(1):21–47, 1999. ISSN 0925-7721. doi: https://doi.org/10.1016/S0925-7721(99)00005-X. URL `https://www.sciencedirect.com/science/article/pii/S092577219900005X`.

P. van Oosterom and B. Meijers. Towards a true vario-scale structure supporting smooth-zoom. In D. Burghardt and M. Sesters, editors, *Proceedings of the 14th Workshop of the ICA Commission on Generalisation and Multiple Representation the ISPRS Commission II/2 Working Group on Multiscale Representation of Spatial Data: Geographic Information on Demand*, pages 1–19, 2011.

P. van Oosterom and B. Meijers. *Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data*, pages 21–42. Nederlandse Commissie voor Geodesie - KNAW, 2012a. ISBN 978-90-6132-339-6.

P. van Oosterom and B. Meijers. True vario-scale maps that support smooth-zoom. In K. de Zeeuw, editor, *S.n.*, pages 1–9. Geospatial World Forum, 2012b. Geospatial World Forum - Theme: Geospatial Industryt and World Economy (Amsterdam, The Netherlands) ; Conference date: 23-04-2012 Through 27-04-2012.

J. W. van Roessel. An algorithm for locating candidate labeling boxes within a polygon. 16(3): 201–209, 1989. doi: 10.1559/152304089783814034.

A. Wolff, L. Knipping, M. van Kreveld, T. Strijk, and P. Agarwal. *A simple and efficient algorithm for high-quality line labeling*, volume 2001-44. Utrecht University: Information and Computing Sciences, uu-cs edition, 2001. ISBN 0924-3275.