***Hygge*bike**:
Finding Copenhagen's Most *'Hyggelige'* Bike Routes

Mads Høgenhaug, Marcus Friis & Morten Pedersen
Geospatial Data Science (Spring 2024)

2024-05-27

Abstract

In Denmark, bicycles and the concept of *"hygge"* are central to the national identity. Leveraging these cultural elements, we develop Hyggebike, a geospatial data science project aimed at identifying Copenhagen's most *hyggelige* (pleasant and cozy) bike routes. By integrating various spatial datasets into a weighted shortest path algorithm, we seek to enhance biking experiences by prioritizing routes that are both practical and *hyggelig*. Using the OpenStreetMap bicycle network in combination with other spatial data, we extract edge-level features from bird observations, tree locations, and road noise datasets, which are used as weights for a weighted shortest path algorithm. The results indicate that the Hyggebike algorithm effectively identifies routes that, while slightly longer, significantly improve the biking experience by incorporating elements that contribute to *hygge*. This approach not only could encourage more people to bike, but could also become a framework for enhancing urban cycling infrastructure by focusing more on the pleasantness of trips. For future work, one could focus on refining the algorithm and expanding its application to other cities.

All code can be found in the project repository: https://github.com/Marcus-Friis/hyggebike

Contents

1 Introduction

Two of the most important aspects of Dane's national identity are bicycles and *hygge*, both often mentioned in descriptions of Denmark (VisitDenmark, 2024) (Nordic Visitor Blog, 2024). According to Copenhagenize (2019), Copenhagen is the worlds most bicycle-friendly city, and a lot of Danes bike as part of their commutes. Despite this, Copenhagen municipality aims for increasing the number of cyclists, with a goal of reaching 50% of all trips to work and education being by bike in 2025 (VisitDenmark, 2022). This goal makes it interesting to investigate the existing bicycle infrastructure. Particularly, we are interested in biking's relationship to *hygge*, and how we can use spatial data to augment pathfinding algorithms to improve the pleasantness of trips. If we can plan more pleasant routes, we can encourage more people to commute by bike.

According to Oxford English Dictionary (2023), *hygge* is defined as *"inspires or engenders feelings of contentment or well-being as from experiencing cosiness, comfort, social harmony, etc.; pleasant, harmonious; cosy, comfortable."*. This term will be frequently used throughout this report. Whenever it is used, it refers to this definition. *Hygge* is used as a label to describe everything that is nice, cozy and pleasant.

This project aims to combine biking and *hygge*; we investigate how we can utilize various spatial data for routing more *hyggelige* paths. Specifically, we develop an algorithm that weighs path information, birds observations, tree locations and noise data to find alternative paths to the traditional shortest path, which instead provides the commuter a more *hyggelig* route, while ensuring the path doesn't become too much longer. Methodologically, we do this by using the bicycle network of Copenhagen from OpenStreetMap, and extracting edge-level *hygge*-features. In this network, nodes are intersections and edges are paths that are accessible by bicycle between each intersection. Thus, the goal of this paper is to collect and extract specific edge-level attributes, which can be used to determine the pleasantness of the individual edge. These edge-level features are subsequently used to develop a weighted shortest path algorithm, that finds best *hygge*-paths.

2 Background

Route planning tools that primarily optimize for the shortest path already exist, and some also optimize for bike lanes, topography, traffic signals, etc. (O'Connor, 2010). Research into similar kinds of route planning optimization also exists; for instance, researchers have optimized for pleasant routes based on Google Street View colors and objects detected (Wakamiya et al., 2019). Others have optimized pedestrian routes based on OpenStreetMap data like green areas, social places, etc., and allowed users to choose their preference (Novack et al., 2018). There is also work on finding routes based on votes for the most beautiful, quiet, and happy streets, using pictures as a reference (Quercia et al., 2014). Analogously, Ribeiro, Mendes (2011) optimizes pathfinding for healthier routes, based on noise and air pollution. These are all papers that deal with a similar pathfinding objective as ours: how can we find the shortest path while maximizing or minimizing external factors?

Effectively, this problem can be seen as a multiobjective shortest path problem (MOSP). While we do not solve the problem with algorithms for such problems, it is interesting to study alternative solutions that solve the problem for inspiration and future research. One implementation of MOSP is "An Improved Multiobjective Shortest Path Algorithm" (Maristany de las Casas et al., 2021), which introduces the Multiobjective Dijkstra Algorithm (MDA). The Multiobjective Dijkstra Algorithm (MDA) is designed to solve the MOSP problem by extending the traditional Dijkstra algorithm to handle multiple objectives simultaneously. Unlike the conventional Dijkstra algorithm, which focuses solely on finding the shortest path based on a single metric (usually distance or travel time), MDA considers additional factors that could influence the final, most optimal route.

3 Data

To plan the most *hyggelige* bike routes through Copenhagen, we gather bike path data and a selection of other data that can be considered to be part of what makes a bike route *hyggelig*. While there are many options for spatial data that influence the pleasantness of a route, we specifically use data about

- Bicycle network

- Bird observations
- Municipal trees
- Road noise

for determining the *hygge* factor. These are not exhaustive factors, and are also highly subjective in terms of their *hygge* factor, yet they function as a demonstration of different spatial data types, which can be used to augment a bicycle network.

The goal of each of these different spatial datasets is to reduce the data to an edge-level feature that can be used in our Hyggebike algorithm, which is a weighted shortest path algorithm. Specifically, we reduce each dataset to a single numerical representation on each edge.

3.1 Data acquisition & description

3.1.1 Bike path data

To create a routing algorithm, we need a network to work on. Particularly, we are interested in a bicycle path network. To get this we use *OpenStreetMap* (OSM). OSM is an open geographic database, which contains data from all over the world. It is maintained by volunteers, where everyone can contribute to the database, so a single point of data is not ensured to be correct, but in aggregate the data can be expected to be accurate enough for this project.

To get the bike paths, we use the library osmnx, which functions as a bridge between OSM and NetworkX. We query OSM with *"Københavns Kommune"* and *"Frederiksberg Kommune"* geocodes. We specify `network_type='bike'` to strictly get bike paths. This results in the network seen in figure 1. In this network, each node is an intersection, and each edge is the path between two intersections. The graph is a multi-directed graph, which means that each edge is directed, and two nodes can have multiple edges between them. It has 25941 nodes and 62439 edges. We are mainly interested in the edges, as the we need to collect edge level features to implement weighted shortest path. All other datasets will in some way be spatially joined with this base network.

In addition to the geographical location of edges, each edge also contains information about the length, max speed, type, number of lanes, and more, although there are a significant number of null values. The most important feature in the OSM graph is the road type of the edge, also referred to as *highway*. The highway contains 17 unique types including *undefined*, with the most frequent types being *residential* (21282), *service* (20848), *tertiary* (7698), *path* (7133), *cycleway* (3006).

3.1.2 Bird observations data

One data source used as a *hygge* factor are bird observations. Specifically, we use data from DOFbasen, provided by Birdlife Denmark (Dansk Ornitologisk Forening, 2024). They provide crowd sourced bird observations with information about species, number of birds, sex, and more. Most importantly, they provide the exact longitude and latitude for observations, allowing us to analyse the point-pattern of bird observations. For this project, we use a dataset with observations between 2020-01-01 and 2024-04-17 in the area of Copenhagen and Frederiksberg. This results in a dataset with $32,710$ bird observations.

As seen in figure 2, bird observations are very clustered, particularly around nature areas. We can validate this by applying clustering techniques and using a clustering evaluation metric. For instance, applying DBSCAN with coarsely tuned parameters leads to a partitioning with 20 clusters, with 82% of points being assigned to a cluster, and a silhouette score of 0.34 for all points and 0.58 when removing all unassigned points. The silhouette score shows significant clustering, especially considering that silhouette score is ill-suited for density-based clusters (Menardi, 2011), which birds are. For more details, see appendix section 7.1.

The reason for birds clustered structure can be due to the mechanism of birds preferring nature and to keep away from traffic, but it is also likely an indicator of the inherent bias of the dataset. Since all observations are crowd sourced, they come from users who willingly report the observation. Users who like to go bird watching and report it to the database are likely going to look for them specifically in areas where they are prevalent, thus reinforcing the inductive bias.

The spatial bias is only further strengthened, as only a small share of users are making these observations, and thus locations where they live and visit will be over-represented. As such, while each point is a valid bird

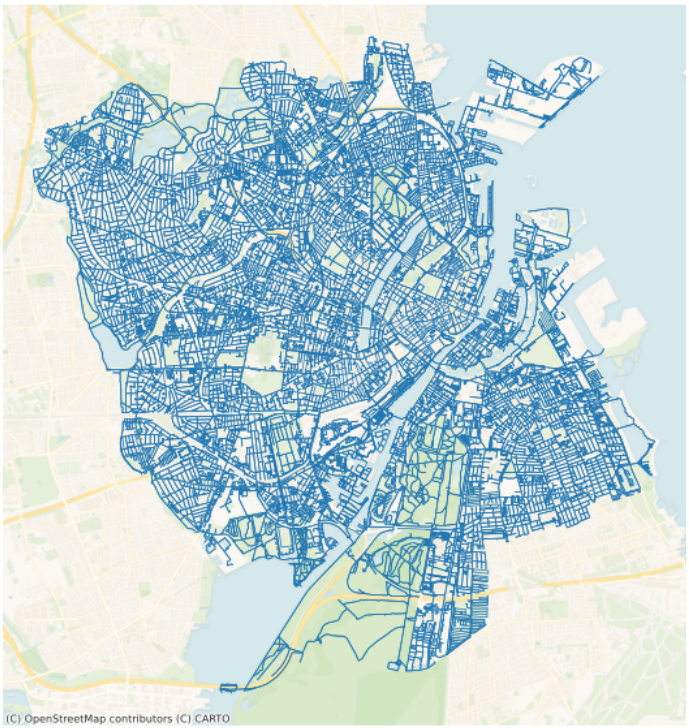


Figure 1: OpenStreetMap bicycle network of Copenhagen and Frederiksberg. The network has 25941 nodes and 62439 edges.

observation, it is essential to be mindful of the bias in the spatial distribution of these observations. Another bias occurs in the kinds of birds that contributors register, as they are probably more likely to register an observation of a rare, interesting bird than a more "boring" bird like a dove. But as this bias of noting an observation might even be nicely correlated with the amount of *hygge* that a bird can be considered to add, it should be fine for the scope of this project.

3.1.3 Tree data

The tree dataset was gathered from from Open Data DK[1], who gets it from Copenhagen Municipality. It contains the 64,505 municipal trees on municipal roads in Copenhagen Municipality, displayed in figure 3. Notably, data from Frederiksberg Municipality is not included, as well as any tree not municipal, meaning locations with many trees (e.g. *Amager Fælled*) seem to have few trees as they are not included. However, trees planted along paths are generally planted by the municipality. This leads to trees being generally clustered in a density-like manner. The dataset includes information on species, year of planting, diameter of crown and trunk and a lot more, but used in this project is the `wkb_geometry` column with the location of each tree.

3.1.4 Road noise data

The third *hygge*-dimension is road/traffic noise. The road noise dataset comes from Copenhagen Municipality. The data is downloadable from Open Data DK[2], and the municipality has presented a map of the data here on their website[3]. The noise levels are not measured with conventional tools, but a calculation based on multiple factors, such as type of road, speed, amount of traffic, etc. The `Nord2000` method of calculation handles this, which is the method the Danish Road Directorate approves of (Kragh et al., 2013). The features from the dataset that we are interested in are:

- **isov1** - These are the noise levels, provided in decibels.
- **geometry** - The dataset's geometry is represented by polygons, with 3,613 polygons in the dataset. This adds some complexity because the roads are represented by linestrings, and they have to be

[1] https://www.opendata.dk/city-of-copenhagen/trae-basis-kommunale-traeer#resource-traer_basis.csv
[2] https://www.opendata.dk/city-of-copenhagen/vejstoej_2022
[3] https://www.kk.dk/borger/affald-og-miljoe/stoej-stoev-og-luft/trafikstoej

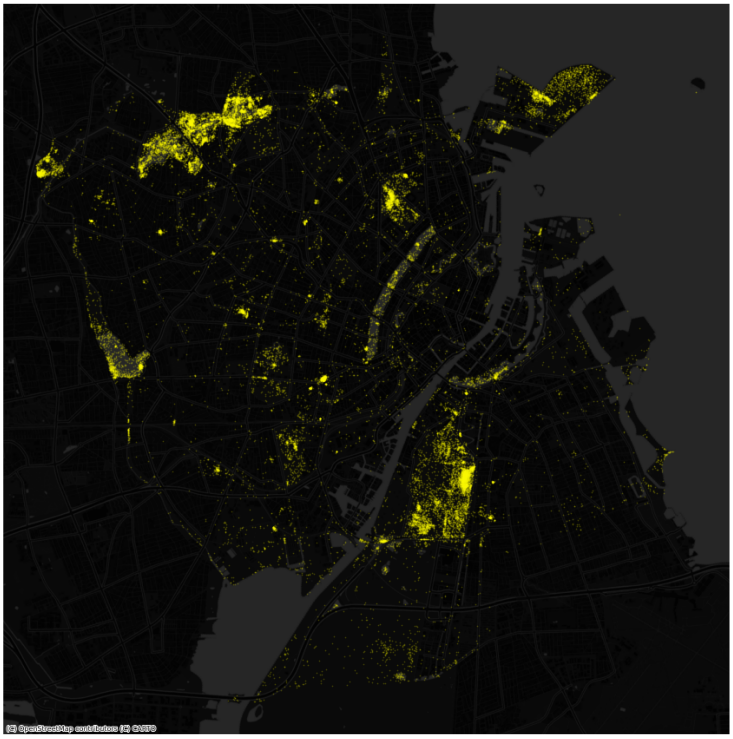


Figure 2: Scatterplot of birds observations in Copenhagen. Each bird observation is a dot on the map. It shows that parks and rivers like *Amager Fælled*, *Utterslev Mose*, *The Copenhagen Lakes* etc. are more densely populated, pointing to bird observations being highly clustered in space.

mapped together. Figure 13 in section 7.2 illustrates this problem, which will be discussed further in section 3.2.3.

3.2 Data processing

All processing is handled by the notebooks:

- `bird-edges.ipynb` - Analysing birds, their clustered nature, and extracting edge level features.
- `tree-edges.ipynb` - Analysing trees, and extracting edge level features.
- `noise-edges.ipynb` - Extracting noise values from polygons to linestrings and predicting the unmapped noise values.
- `hyggefinder.ipynb` - Master notebook, combining all the individual data sources to analyse, visualize and explore the notion of *hygge* in the bicycle network, along with creating the Hyggebike algorithm.

The goal of all our data processing boils down to extracting edge level features, that can be used to weight the length of a path, and thus influence the shortest path between two nodes.

To process all of the data, we require multiple geospatial tools. To this end, we use the containerized environment `'gds_py'`, which is *"a container providing a fully working Jupyter Lab installation, additionally loaded with a comprehensive list of geospatial python packages."* (Arribas-Bel, 2022, 2024). Specifically, we use Docker image `gds_py:8.0` for all code. With this container, we have all the tools needed for the tasks at hand, in a working, reproducible environment.

When working with geospatial data, it's important to choose an appropriate coordinate reference system (CRS) for the country in which the geospatial operations takes place. Since all our data only is in Denmark, we use `EPSG:28532` as the CRS for all processing. This CRS uses meters as units which makes it more clear what the distances and spatial relationships represent in a real-world setting (EPSG.io, 2024). Throughout this project, whenever we work with any sort of geospatial data, the data is projected to the this CRS.

Figure 3: Scatterplot of trees in Copenhagen Municipality. Each tree is a dot on the map. It shows that parks and graveyards like *Fælledparken* and *Assistens Kirkegård* are more densely populated. We also see that many paths either have many trees or (close to) none.

3.2.1 Bird data processing

All bird data is processed in `bird-edges.ipynb` notebook. There are many different applicable methods to process this data; our approach involves using intersecting bird buffers. To this end, we create a 50-meter buffer around each bird observation and count the number of bird buffers that intersect with each edge. These intersection calculations are performed using a spatial join. A spatial join is an operation that combines two datasets based on their spatial relationship. In this context, the 50-meter buffer ensures that bird observation points are joined with nearby edges. To count the number of birds per edge, we group each edge together in the spatially joined dataset. The second to last step is joining the newly acquired bird counts back to the original edges. Finally, we normalize the bird count by path length, since some paths are longer than others. The resulting operation leads to a scheme, where each observation can be assigned to multiple edges, or none at all. The 50-meter buffer accounts for birds being visible in a decent area around them, and for being generous regarding any minor errors in the crowd sourced data. The resulting distribution of the edge-level bird feature is highly skewed, with over half of the edges having 0 birds.

3.2.2 Tree data processing

All tree data is processed in `tree-edges.ipynb` notebook. Each tree is mapped to its nearest edge. This is achieved using osmnx's `'nearest_edges'` function, which uses an R-tree spatial index. Spatial indexing is a technique to efficiently store and retrieve data based on their geographic location. Using a spatial index for this processing part is instrumental, as the alternative is iteratively calculating the distance between a point (tree) and *all* the edges (paths). With $64,505$ points and $\sim 62,000$ edges, the computational complexity to exhaustively calculate all the distances would simple be infeasible. We map a tree to an edge, if the tree is within 10 meters of the edge. Then, for each edge we sum the number of trees, and normalize the tree counts by length of the path, similar to that in section 3.2.1. We use a 10-meter distance to ensure trees are close enough to be relevant during a bike ride. This avoids counting trees inside parks or distant areas that do not enhance the riding experience.

Figure 4: Representation of raw noise data using polygons. Highways are marked as the highest noise contributors, while nature areas are the quietest.

3.2.3 Noise data processing

All noise data is processed in `noise-edges.ipynb` notebook. A buffer of 10 meters is created around each road to identify which roads intersect with the nearest polygon(s). Due to the size of both datasets, and the spatial operations needed, we create a spatial index on the polygon, using Geopandas' `sindex` function (developers., 2024). By using spatial indexing, we can quickly query and analyze which linestrings intersects with which polygons based on their location, without having to scan through large amounts of data to find the relevant information. After establishing the buffer zones around the roads, we proceed to identify intersections between the buffered roads and the noise polygons. This is achieved by iterating over each road in the dataset and leveraging the spatial index to find potential matches from the noise polygons dataset. For each road, we first determine the possible matches by querying the spatial index with the bounding box of the road's (buffered) geometry. Each potential match is then checked for an actual intersection using the intersects method, and if an intersection is found, we calculate the intersection area. Since each road can intersect with multiple polygons, we want to obtain the expected noise from all the polygons a linestring might intersect with. This is achieved by first grouping the intersection results by road ID and summing the intersection areas. Then we obtain a intersection percentage by dividing the intersection area for each polygon, with the total intersection area a linestring has with the nearest polygon(s). To derive an expected noise output for each road segment, we multiply the noise level of each polygon by the normalized intersection percentage, producing a weighted noise value. These weighted noise values are then summed for each road segment to obtain the final weighted noise value for each linestring. Finally, we merge the new results back into the original road dataset, giving us a weighted noise value for each road. The distribution of decibel levels can bee seen in figure 14 in section 7.2.

When extracting the noise values to a linestring basis, the noise values from these big polygons will be mapped to hundreds of nearby roads, leading to a rise in the count of the polygon's noise values.

There are about $16,000$ rows in the dataset that do no get assigned a weighted noise value. This happens when roads are more than 10 meters (the buffer zone) from the nearest polygon. All roads in Frederiksberg are missing noise values, because the noise data set is produced by Copenhagen municipality, and thus does not contain any information about Frederiksberg. There are also some roads in Copenhagen municipality that are lacking a weighted noise value. In figure 6a we see which roads of Copenhagen that have had a

Figure 5: This polygon (index 197 in the dataset) practically covers all the large roads in Copenhagen municipality. This polygon has an area of $4.771.942m^2$. The noise level for this polygon is 65 decibels.

noise value mapped to it. To account for this, we implement a simple model to predict the noise values for the missing roads.

3.2.4 Modelling noise values

We have 45, 869 rows with a non-zero value in the weighted noise column. With `osmnx` we get some additional meta data about the roads, such as `length`, `number of lanes`, `maxspeed`, and `type of highway`[4]. These features will be used to train a model for imputing missing values. The task of getting the data ready for the model is mostly handled by `pandas` and `scikit-learn`, and given the lack of geospatial operations for this, we wont dive that deep into the process in this report. For a detailed walk-through of the data preparation process, we refer to the accompanying notebook. We have implemented a basic Random Forest Regressor to predict noise values based on the given features and the network property edge betweenness. The resulting model has an r^2 score of 0.55, indicating moderate performance. Although this score is not ideal, it is sufficient for the prototypical processing of *hygge* data in this project. Future improvements should focus on enhancing the model by incorporating more relevant features and possibly including neighborhood information to increase spatial awareness. The final map displaying both the original and predicted data can be seen in figure 6b.

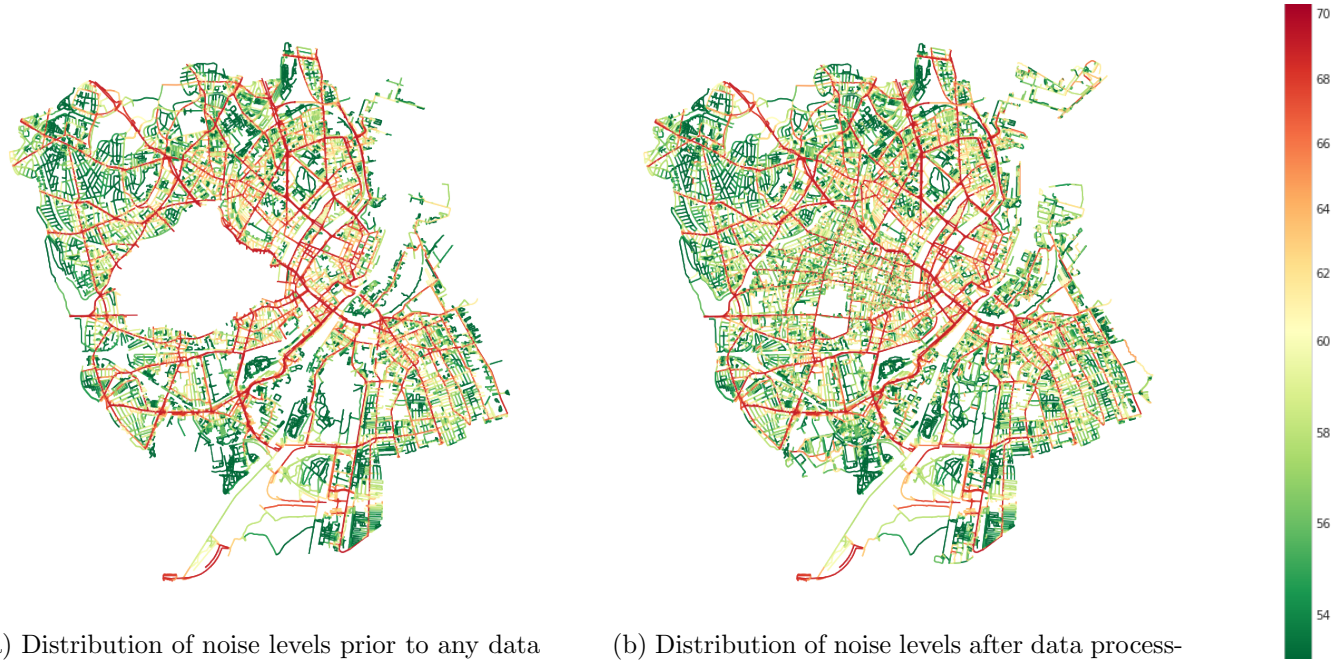
3.2.5 Challenges working with the noise data

One of the main challenges during this noise processing is ensuring accurate and efficient spatial operations. Creating buffers and performing intersection analysis can be computationally intensive, especially with large datasets. Choosing the right buffer zone for the linestrings is experimental and involves a trade-off between the number of linestrings mapped to a polygon and the quality of the `weighted_noise` feature. Larger buffer zones ensure that more linestrings are mapped to a polygon, but this can result in possibly too many polygons being mapped to a linestring. Consequently, the `weighted_noise` feature may include more than just the nearest polygons. For a large road in Copenhagen, this means there is a possibility that smaller roads affect the calculated noise resulting in a lower noise level than expected, when weighting the noise levels from the various polygons a linestring has intersected with.

3.3 Pathfinding algorithm

The pathfinding algorithm is created and explored in `hyggefinder.ipynb` notebook. While there are many applicable approaches for this sort of specialized shortest path algorithm, we demonstrate one of the simplest implementations. The implemented approach categorizes each feature, and boosts or penalizes the path length according to a manually set factor, based on how *hyggelig* we deem the feature to be. For instance,

[4]The notebook has all the features listed in the `features` variable. Available at https://github.com/Marcus-Friis/hyggebike/blob/main/notebooks/noise-edges.ipynb



(a) Distribution of noise levels prior to any data processing

(b) Distribution of noise levels after data processing

Figure 6: Figure 6a shows the linestrings that have been mapped to a nearest polygon. Figure 6b shows the final noise map, with the predicted values added. The legend shows the noise in decibel.

if a path contains a lot of bird observations, we shorten the length of the path to encourage bikers to use it. However, if there is a lot of traffic noise at a path, we lengthen the path by some factor to discourage it. Using Dijkstra's shortest path algorithm, we can apply it to the newly weighed path length, the *hygge length*, if a route is pleasant enough, it will become shorter than the shortest path in terms of *hygge* length. The actual weighing is implemented as

$$\hat{d}_{hygge} = d \prod_{i=1}^{|w|} w_i$$

As for the actual weights, birds are weighted at 0.8, 0.9, and 1, for the $0 \rightarrow 80th$-, $80 \rightarrow 90th$-, and $90 \rightarrow 100th$ percentile respectively. Trees are weighted at 0.8 and 1 for the $0 \rightarrow 90th-$ and $90 \rightarrow 100th$ percentile respectively. Noise is divided in to 5 equally large percentile groups, and weighted at 0.8, 0.9, 1, 1.1, and 1.2. The OSM data for type of highway is qualitatively ranked and weighted between 0.8 and 1.2. Experimenting with different ranges of values we found the range of around $0.8 - 1.2$ to nicely value both distance and *hygge* factor, as it created routes that we would not deem too much longer, but often decently more *hyggeligere*. The theoretical lower bound for a possible weight, if all *hygge* features were to be maxed out, is a weight of $0.8^4 = 0.41$, meaning that if a path were to be maximum *hygge*, it will be more than halved in length when calculating *hygge* length. Conversely, if a path is the most anti-*hygge* path, it will be penalized with a weight $1.2^2 = 1.44$ i.e. it will be extended with almost half its length. The weights are manually defined, and they could be tuned using other weights based on the commuters willingness to bike greater or lesser distances.

4 Results

The main result of this paper is the Hyggebike pathfinding algorithm. We have applied it and looked into the properties of the reweighed network based on the *hygge* weights, which we present in this section.

4.1 Finding the best *hygge* paths

When using the *hygge* algorithm, we can inspect its effect on individual routes. For instance, on a sample of 5 routes, we compute the standard shortest path weighed with path length, and the new *hygge* path, weighed with the *hygge* length. Both routes are shown for all 5 example routes in figure 7, with the standard shortest path route being marked with red, and the new *hygge* path with green.

Figure 7: A selection of shortest routes (red) and their complimentary *hygge* route (green).

Nodes		***Hygge* route**		**Shortest route**		**Index**	
Origin	Destination	$\hat{d}$	d	$\hat{d}$	d	*Hygge*	Reroute
1632453764	8089128	5257	6632	6922	6084	0.76	1.09
298629942	6117851027	3526	4528	4243	4174	0.83	1.08
3903388174	976512194	3914	5552	5188	4708	0.75	1.18
5434288601	450792743	6764	8312	8498	8220	0.8	1.01
6551516283	11806513465	2873	3902	3689	3073	0.78	1.27

Table 1: Route Comparison of *Hygge* and Shortest by length, including and index which is the *hygge*-length of the *hygge* route divided by the *hygge* length for the shortest route, and the same for the actual length, for the *hygge* index and reroute index respectively. d denotes the actual length of a path, and $\hat{d}$ denotes the length in the weighted *hygge* space.

From this figure, we can see how the *hygge* route is usually a bit longer, but with a little knowledge of Copenhagen, it is evident that routes are prioritised in areas with nature or where there are dedicated cycleways. This is further confirmed by table 1. Here, we see the reroute index, which is what we call the ratio between path length of the *hygge* path d_{hygge} and the shortest path $d_{shortest}$.

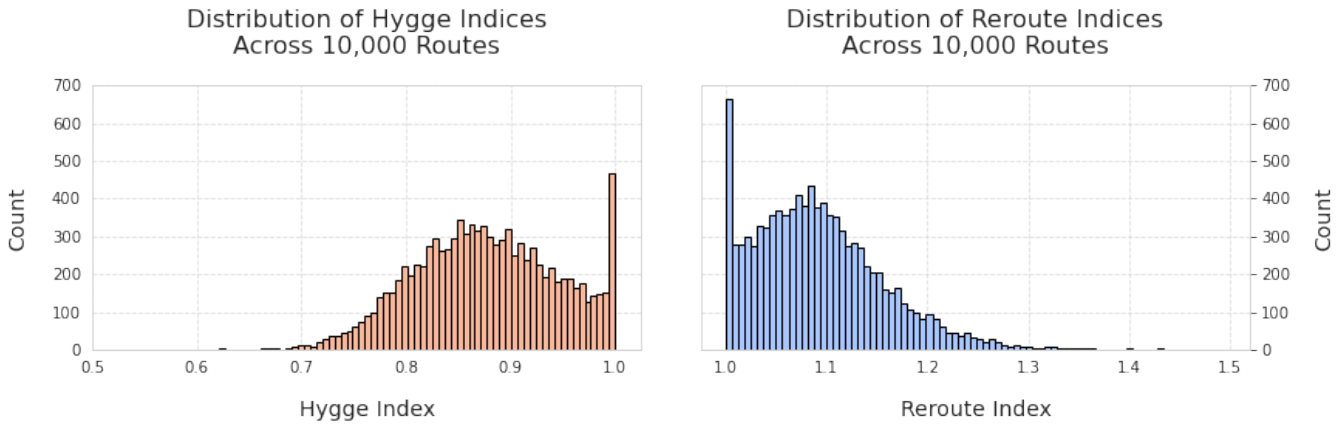
$$reroute = d_{hygge}/d_{shortest}$$

From the reroute index, we see that these paths are between 1.01 and 1.27 times longer than the original path. However, the *hygge* index, which is similarly the ratio between the *hygge*-length of the *hygge*- and the shortest path, ranges between 0.75 and 0.83.

Scaling this experiment up to $10,000$ random origin-destination pairs, we can compute the distribution of *hygge*- and reroute index for the network, as seen in figure 8. From this, it is evident that a significant portion of paths remain unchanged. However, many paths are slightly rerouted but gain significantly more *hygge*. This is demonstrated by the means of the two distributions, 0.874 and 1.094. These values indicate that the average path length is extended by a factor of 1.094, yet the increase in *hygge* is proportionally larger compared to the loss of *hygge* when taking the shortest path..

4.2 Aggregate routes

From the reweighed edge lengths, we can inspect the effect of *hygge* on the overall network. To start, we visualize the spatial distribution of *hygge* edges for the 4 different *hygge* features in figure 9. We see that



(a) *Hygge* Index Distribution

(b) Reroute Index Distribution

Figure 8: Comparison of Reroute and *Hygge* Index Distributions

each feature conforms to common intuition; bird weights are primarily low in nature areas and clustered around other low weighted edges, as expected due to birds clustered nature. Trees weights are harder to interpret, yet it seems to conform to contiguity-based clusters, with longer stretches and local areas boosting the *hygge*-length. Noise weights show properties of larger, more central paths being prone to being noisy, whereas more suburban paths and paths in nature are more quiet. Lastly, highway weights show that most larger roads share the same highway type. and are thus penalized, whereas smaller roads and paths that are unavailable for cars are encouraged, similarly to the traffic noise weights.

Taking the product of all the 4 feature weights from figure 9 results in the map seen in figure 10. This map shows properties of all 4 composite maps, with larger trafficked highways being discouraged due to noise, highway type, lack of birds and trees. Meanwhile, nature areas and dedicated bike paths are strongly encouraged, and smaller residential streets are also preferred over larger roads.

Strictly visualizing the edge weights does not tell us the impact on pathfinding in the network though. Instead, we use betweenness centrality to visualize each path's importance in shortest path routing. Specifically, we compute the betweenness centrality for both regular shortest path with edge length as weight, and for *hygge* paths with *hygge* length as weight. Doing so results in figures 11a and 11b. These figures show the most frequently used paths in route planning with respect to their weight. Comparing them, we see that the most important edges in the network shift as we increase the focus on finding the most pleasant routes. For instance, *Langebro* is a bridge that connects *Amager* with the rest of Copenhagen, and is among the top 10 highest betweenness scores in the graph. However, when calculating the *hygge* betweenness, *Langebro* is not even in the top 4000 most important edges, and *Lille Langebro* takes its place as the $10th$ most important edge. This makes sense, as *Lille Langebro* is a dedicated bike path with much less traffic noise, and is not too much of a detour. We see the same thing happening in other areas, e.g. *Den Grønne Sti* takes the place of many neighboring roads, like *Nordre Fasanvej* and *Falkoner Allé*. And the paths around *Damhussøen* is prioritised over the much less *hyggelig Ålekistevej*.

All of these differences between the two betweenness definitions are summarised in figure 11c. This visualization shows the difference between the *hygge*- and regular betweenness. It uses a diverging color palette to emphasize which paths gain or lose importance. All of the aforementioned differences between the betweenness plots are more clearly visible here, summarising how the network routing changes with the introduction of *hygge*.

(a) Bird weights

(b) Tree weights

(c) Noise weights (with predicted values)

(d) Highway weights

Figure 9: Map visualizing how each edge is affected by the assigned edge weight for each feature. All maps share the same color range, making it easier to compare how each area of the map is affected.

Figure 10: Composite *hygge*-score of all paths, created by multiplying all edge weights seen in figure 9. From this map, we can tell how each individual edge is weighted. The general pattern is areas of nature being encouraged through lower length weights, while more centralized paths are discouraged, primarily due to traffic noise and highway type.

(a) Betweenness centrality weighted by path length. Shows the traditionally most important edges in shortest path routing.

(b) Betweenness centrality weighted by *hygge* length. Shows the most important edges in *hygge* bike path routing.

(c) Difference between path length betweenness centrality and *hygge* length betweenness centrality. It is colored with a diverging color palette to showcase the change in path importance when going from shortest path routing to *hygge* path routing.

Figure 11: 3 betweenness centrality maps, showcasing how network routing changes when transitioning from traditional shortest path routing to *hygge* path routing.

5 Discussion

While our approach is a step forward in combining cultural and environmental factors into bicycle route planning, it does have some limitations. To address these, we need ongoing data collection, better predictive models, and improved user customization to make sure *hygge* optimization is both accurate and personally relevant.

The largest shortcoming is the missing data of Frederiksberg. Both the noise- and tree dataset are lacking data for this area. For the noise dataset, we circumvented this by implementing a simple model that predicts a noise value based on multiple characteristics of a road. The predictive model, while useful, lacks the precision and detail of direct measurements or authoritative calculations, such as those produced by the Nord2000 method used by the Danish Road Directorate. Due to the lower quality of data for Frederiksberg, the noise values might be less accurate. This could affect the overall effectiveness and reliability of our *hyggelige* route recommendations in this area. Users might notice that the routes don't align as well with the intended noise level reduction, which could impact the *hygge* factor.

For extending the framework to other cities, if similar data is available, the method can be applied. Maybe cultural factors should be taken into account, as what is considered *hygge* is not the same. This could also be intentional as it could be marketed as the Danish way of biking. The data would have to be updated as areas change. Also it could be updated depending on season, many birds migrate and trees are not equally beautiful year round. Qualitatively the *hygge* routes seem very representative of what we would suggest as an alternative route, even with just the few features that are by no means a perfect representation of *hygge*. But it would seem that these features correlate with other *hygge* features not in the dataset, like going along a body of water or avoiding traffic lights, which are due to e.g. birds being around water, and traffic lights probably correlating with traffic noise. The method would likely transfer well to cities similar to Copenhagen, but could need tweaking for cities with very few birds and trees, but with other *hygge* factors like cafés and street lights.

6 Conclusion

We have created *hygge*bike; a pathfinding algorithm for Copenhagen's bicycle network that prioritizes paths that are *hyggelige*, pleasant and cozy. It is a shortest-path-algorithm-based approach, where weights are assigned to individual paths of the bicycle network using a variety of path-level features. These weights either encourage usage through shortening the path length, or penalize paths that are unpleasant through lengthening them, resulting in the so called *hygge* length. The edge-level features are extracted from various geospatial data sources, leveraging spatial operations for mapping the data to individual edges. Specifically, we use bird observations, municipal trees, noise data, and road type for deeming the *hygge* factor of a road. The result is a reweighed bicycle network, on which shortest paths on the *hygge* length lead to the most pleasant routes. With this very simple approach, we achieve surprisingly effective *hygge* routes, showing that example routes are rerouted in a sophisticated manner that, when using personal intuition about the streets of Copenhagen, make a lot of sense. We show the aggregate effect of the *hygge* paths compared to the traditional shortest path, and visualize how the flow in the network changes as shortest path prioritization changes based on the betweenness centrality.

6.1 Future work

The project as-is is heavily missing some actual evaluation metric for how *hygge* a route or street is. Figuring this out would facilitate a quantitative understanding of how effective the *hygge*bike algorithm is, and thus which areas of the algorithm need to be improved. Furthermore, the algorithm relies on a manually set weight-based approach, which, while effective to a certain extent, lacks the robustness and flexibility of more advanced algorithms. Further investigations should focus on shifting from this manually set weight-based approach to a multiobjective shortest path algorithm like the MDA outlined in the background. This change would likely enhance efficiency and accuracy by considering multiple optimization criteria. Future work should include integrating this algorithm and testing it across various scenarios. Additionally, we intended to implement a website for our project using Streamlit. However, due to time constraints, we were unable to complete this. Future efforts should prioritize developing a user-friendly web interface to enhance accessibility and user experience, allowing for easier interaction with our *hygge* pathfinder algorithm.

References

Arribas-Bel Dani. Dockerhub: gds_py, tag 8.0. 2022.

Arribas-Bel Dani. Python: gds_py. 2024.

Copenhagenize . 2019 Copenhagenize Index. 2019. Last accessed: 2024-05-25.

Dansk Ornitologisk Forening . DOFBasen. 2024. Last accessed: 2024-04-17.

EPSG.io . EPSG:25832. 2024. Last accessed: 2024-05-23.

Kragh Jørgen, Michelsen Lene Nøhr, Fryd Jakob, Jakobsen Jørgen. Nord 2000 handbook. June 2013. Copyright Vejdirektoratet 2013.

Maristany de las Casas Pedro, Sedeño-Noda Antonio, Borndörfer Ralf. An Improved Multiobjective Shortest Path Algorithm // Computers Operations Research. 2021. 135. 105424.

Menardi G. Density-based Silhouette diagnostics for clustering methods // Statistics and Computing. 2011. 21. 295–308.

Nordic Visitor Blog . 10 Cool Facts About Denmark. 2024. Last accessed: 2024-05-25.

Novack Tessio, Wang Zhiyong, Zipf Alexander. A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data // Sensors. 2018. 18, 11.

O'Connor Mary Catherine. Google Maps Finally Adds Bike Routes. 2010.

Oxford English Dictionary . hygge (adj.). July 2023. Last accessed: 2024-05-25.

Quercia Daniele, Schifanella Rossano, Aiello Luca Maria. The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. 2014.

Ribeiro Paulo, Mendes José. Route planning for soft modes of transport–Healthy routes. 06 2011.

VisitDenmark . Facts and Figures on Cycling in Denmark. 2022. Last accessed: 2024-05-25.

VisitDenmark . Fun facts about Denmark. 2024. Last accessed: 2024-05-25.

Wakamiya Shoko, Siriaraya Panote, Zhang Yihong, Kawai Yukiko, Aramaki Eiji, Jatowt Adam. Pleasant Route Suggestion based on Color and Object Rates // Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. New York, NY, USA: Association for Computing Machinery, 2019. 786–789. (WSDM '19).

developers. GeoPandas. GeoDataFrame.sindex. 2024. Last accessed: 2024-05-23.

7 Appendix

7.1 Appendix A: Clustering birds with DBSCAN

To validate the clustered structure of birds, we apply DBSCAN. DBSCAN is used since birds seem to be clustered with a density-based structure. We use *scikit-learn's* implementation of DBSCAN with parameters `eps=200` and `min_samples=100`. Doing so yields the clusters seen in figure 12; a partitioning with 20 clusters, with 82% of points being assigned to a cluster, and a silhouette score of 0.34 for all points and 0.58 for all assigned points.

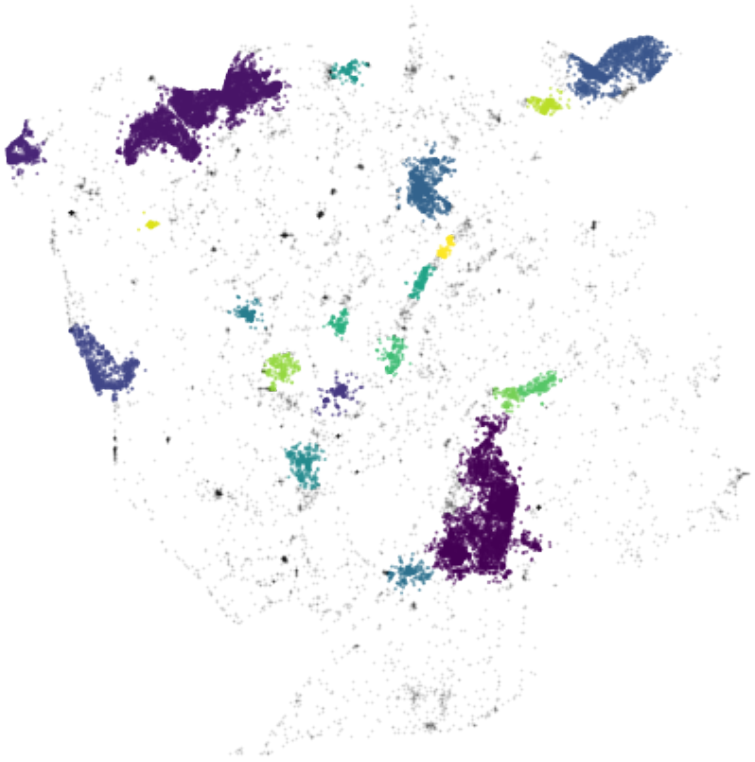


Figure 12: Scatterplot with bird observations colored by cluster, showcasing the density-based structure of bird observations. There are 20 clusters, so be mindful that some clusters can look like they share a color without being the same cluster.

7.2 Appendix B: Accompanying road noise visualizations and information

The reason for the abrupt change in distribution for the higher levels of decibels, as seen in figure 14, comes down to the fact that a few polygons covers all the large roads in Copenhagen. This is evident in figure 5, which is just a single polygon. This is the biggest polygon in the dataset, and it has an area of $4.771.942m^2$. For comparison, the median area of the polygons are $770m^2$
test: 14

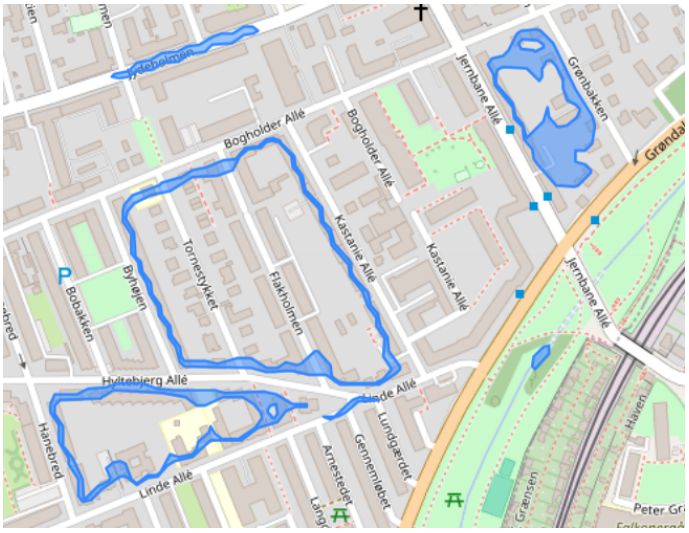
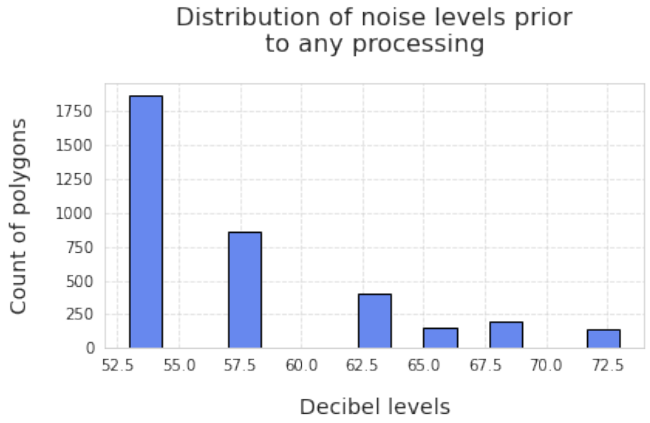
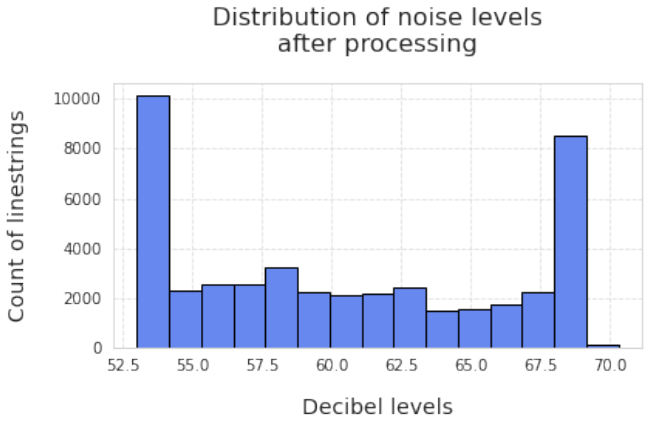


Figure 13: Example of 6 polygons from the noise dataset. This image demonstrates the challenge of calculating intersections between roads and polygons. Without including a buffer zone, many polygons fail to intersect with roads, as they often encompass non-road objects like apartment blocks.



(a) Distribution of noise levels prior to any data processing



(b) Distribution of noise levels after data processing

Figure 14: At a first glance the two distributions vary a lot, especially for the high decibel levels. However, recall that a single polygon can cover a lot of roads in Copenhagen, see figure 5 for example. That single polygon has been mapped to hundreds of roads, and thus the count of these decibel levels naturally increase as well

7.3 Appendix C: Meta data tables

	Software metadata description	**Details**
S1	Current software version	Docker image `darribas/gds_py`, with the tag `8.0`, which runs Python 3.9.12
S2	Permanent link to your code in your Github repository	https://github.com/Marcus-Friis/hyggebike
S3	Legal Software License	APACHE LICENSE VERSION 2.0
S4	Computing platform / Operating System	Pop!_OS 22.04 LTS, Windows 10, Windows 11
S5	Installation requirements & dependencies for software not used in class	
S6	If available, link to software documentation for special software	We did not use any software outside of the gds_py image.
S7	Support email for questions	mahf@itu.dk; mkrh@itu.dk; mohp@itu.dk

Table 2: Software metadata

	Data metadata description	**Details**
D1	Data license	**our data**: Apache License 2.0 **noise data:** CC0 1.0 **bird data**: custom license: https://dofbasen.dk/rights.php **tree data**: Creative Commons Attribution 4.0
D2	Dataset name / main properties	**noise data**: polygon geometry, used attributes: 'ID', 'isov1', 'geometry'. The dataset also contains a column named 'isov2'. However, we couldn't find information which of these are the "correct" feature to use. They are both decibel levels, and vary only very little. Therefore, we use 'isov1' throughout the project. Generated using the `Nord2000 model`, Source: https://www.opendata.dk/city-of-copenhagen/vejstoej_2022 **tree data**: used attributes: 'wkb_geometry' **bird data**: used attributes: 'obs_laengdegrad', 'obs_breddegrad'

Table 3: Data metadata

7.4 Appendix D: Contribution statement

All members have contributed equally.

7.5 Appendix E: Use of generative AI

We have used OpenAI's ChatGPT (version 3.5 and 4o) as well as Google's Gemini (Gemini vanilla; the free version). They have primarily been used as 'consultants' to help with various code errors, LaTeX formatting, and summarizing relevant research papers. Moreover, ChatGPT has a custom GPT named "Consensus" which can find relevant research papers based on the prompt. It should be mentioned, that we never follow a generative AI tool blindly. All its claims have to be fact checked. However, what these tools really excel at, is to point us in the right direction when searching for various information. OpenAI's Dalle-3 model generated the frontpage image.