Processing the World with H3 and PostGIS

Darafei Praliaskouski

PostGIS Project Steering Committee member
OpenStreetMap contributor

Founder at Maumap.com Geospatial consulting
Previously at Kontur.io

OpenAerialMap / OpenDroneMap fan
Lately Meshtastic LoRa radio planning

Living in Batumi, Georgia

Maumap


🇬🇪 Georgia has mountains

Mountains are fun for radio

Telecom towers are always on top.

You can put yours nearby.

Meshtastic

An open source, off-grid, decentralized, mesh network built to run on affordable, low-power devices.

https://meshtastic.org/

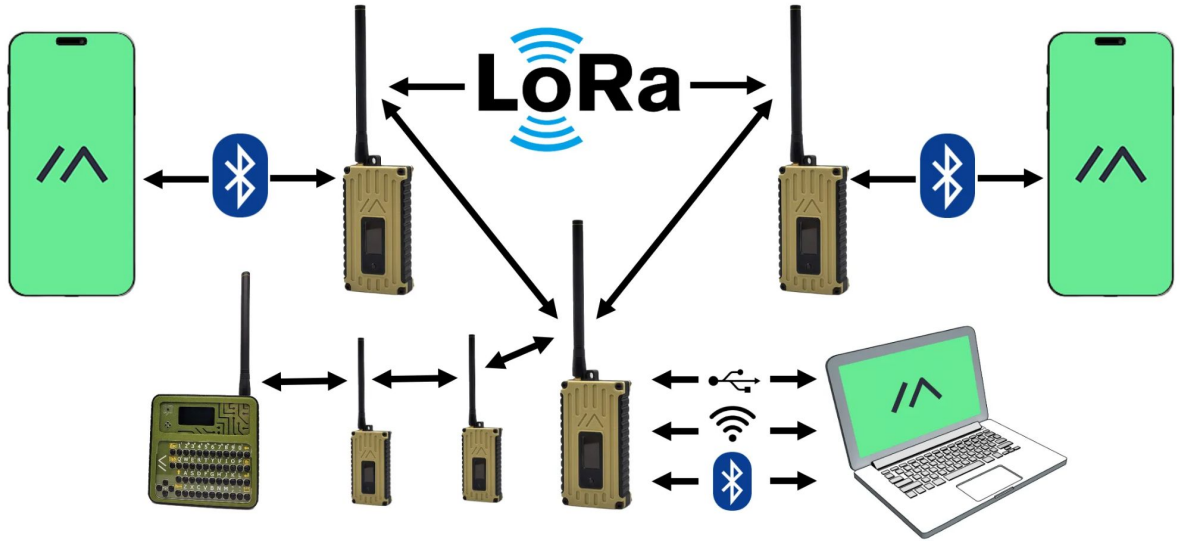
Runs on LoRa,

which you can meet in IoT

Devices as low as $12…

Why not put them

every**where**?



What we learned deploying our LoRa network

- Surprisingly long range on direct visibility - we have 56km links.
- Still works in urban setting. More people, more noise, more volunteers, more repeaters.
- It’s easy to put a solar powered node on the pole on reasonable height.
- Largest issue is logistics of getting node there and picking a spot where it’s useful.
- There are people who climb mountains for fun and don’t mind putting your node there.
- Having free direct messaging with other engineers is fun :)

So, today we’ll look at the question - **where** do we put our nodes?

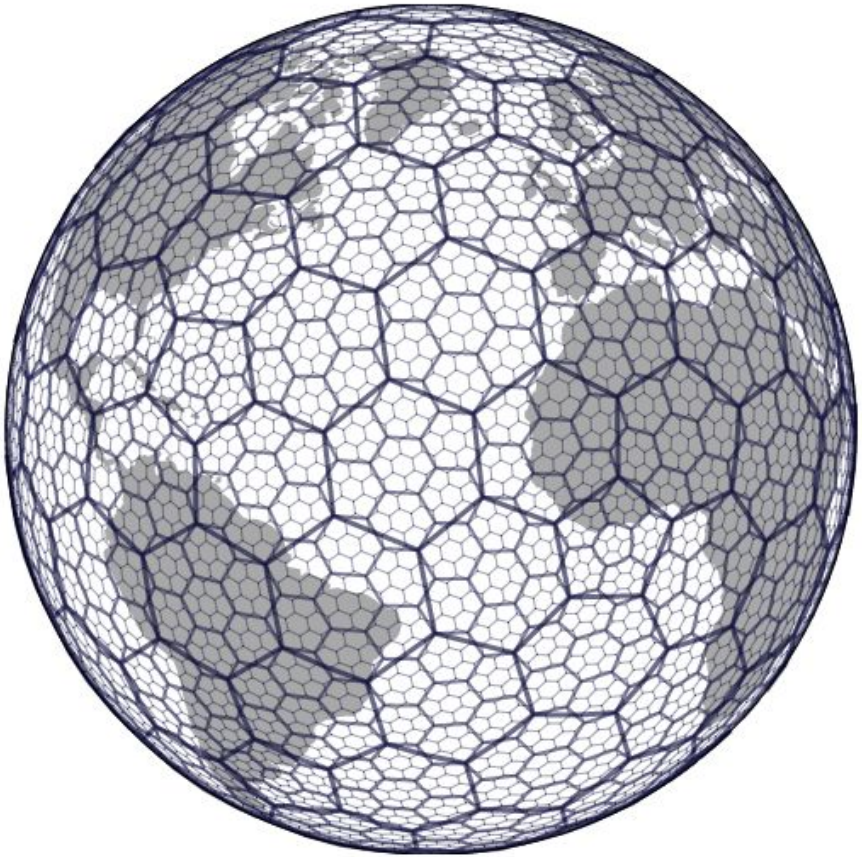
H3

Hierarchical Geographical Index

Mostly hexagonal

Useful for statistical analysis

Optimized for analyst’s mental health



In the H3 indexing system, how many edges does a hexagon have on average?

At resolution 1, average hexagon has **6.41** edges

At the odd H3 resolutions some hexagons get extra **distortion vertices** when they belong to multiple icosahedral faces

In addition to **560** hexagons, resolution 1 have:

- **240** heptagons
- **30** octagons
- **12** decagons

KONTUR

KONTUR

Pentagons in H3 hexagonal grid. All of them are centered on water +

#30DayMapChallenge

Adventures of Polygons traveling between different Earth models

Flat sheet🗺 to Globe🌐: points you may expect inside are now outside when the segments follow great circle rather than rhumb lines.

Globe🌐 to Flat sheet🗺 to Cylinder🛢: the polygon is now inside out on every rotation of the cylinder.

Flat sheet🗺 to Cylinder🛢: edges of a flat map can sometimes be seen as a boundary inside the polygon

h3_pg

Best set of extensions for supporting h3 in Postgres.

https://github.com/zachasme/h3-pg

Created and maintained by Zacharias Knudsen

Thanks a lot for accepting our pull requests :)

Start by choosing your uncertainty

$$\sigma_x \sigma_p \geq \frac{\hbar}{2}$$

Most of my projects are in one of two resolutions:

- Resolution 8:
  - “walking distance”
  - ~400m edge length
  - the planet will likely fit into your laptop
- Resolution 11:
  - “specific building”
  - ~ 30m edge length
  - can use H3 for pathfinding

If you think you need more detail, likely H3 isn’t the best tool for your task.

Formalizing task

I’d love to **find locations** for growing a network of radio repeaters in **Georgia**.

I want every next one to **grow** the network towards new **populated area**.

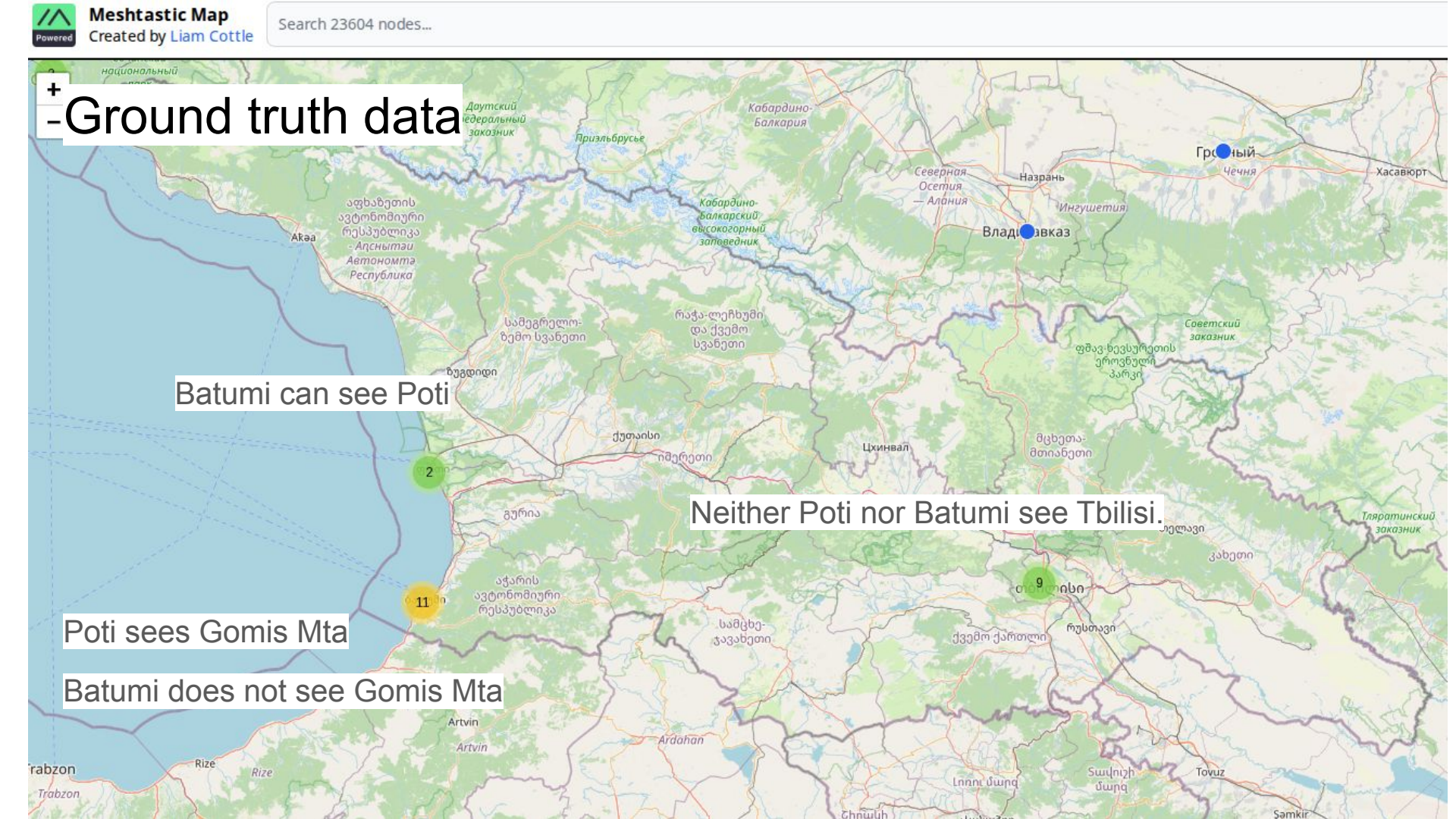
Resolution 8 should be okay for first pass.

Let’s assume that 60 km links in **line of sight** are working every time.

DEM will be enough - we’re not looking closely at cities this time.

I want to get to the installation site **by car**.

Model we build shouldn’t contradict existing **ground truth** cases we observe.


Meshtastic Map
Created by Liam Cottle
Search 23604 nodes...
Ground truth data
Batumi can see Poti
Neither Poti nor Batumi see Tbilisi.
Poti sees Gomis Mta
Batumi does not see Gomis Mta
2
11
9
Грозный
Назрань
Владикавказ
Чечня
Ингушетия
Хасавюрт
Цхинвал
Кабардино-Балкария
Северная Осетия — Алания
Кабардино-Балкарский высокогорный заповедник
Призельбрусье
Советский заказник
Тляратинский заказник
Artvin
Rize
Trabzon
Ardahan
Tovuz

What data will we need?

Some existing nodes

Elevation raster

Roads

Country boundary

Population

What data will we need?

Some existing nodes	points
Elevation raster	raster
Roads	lines
Country boundary	polygon
Population	h3 grid

What data will we need?

Some existing nodes	points	sketch in geojson.io
Elevation raster	raster	GEBCO 400m global grid
Roads	lines	OpenStreetMap
Country boundary	polygon	OpenStreetMap
Population	h3 grid	Kontur Population

GeoJSON points to H3

```
$ ogr2ogr -f PostgreSQL PG:"" data/in/existing_mesh_nodes.geojson -nln
mesh_initial_nodes -nlt POINT -lco GEOMETRY_NAME=geom -overwrite -a_srs
EPSG:4326
```

```
-- Create seed towers table projected to H3 resolution 8

create table mesh_initial_nodes_h3_r8 as

select
    h3_latlng_to_cell(geom, 8) as h3,
    string_agg(coalesce(name, 'seed'), ‘, ’) as name,
    ST_Collect(geom) as geom
from mesh_initial_nodes
group by 1;
```

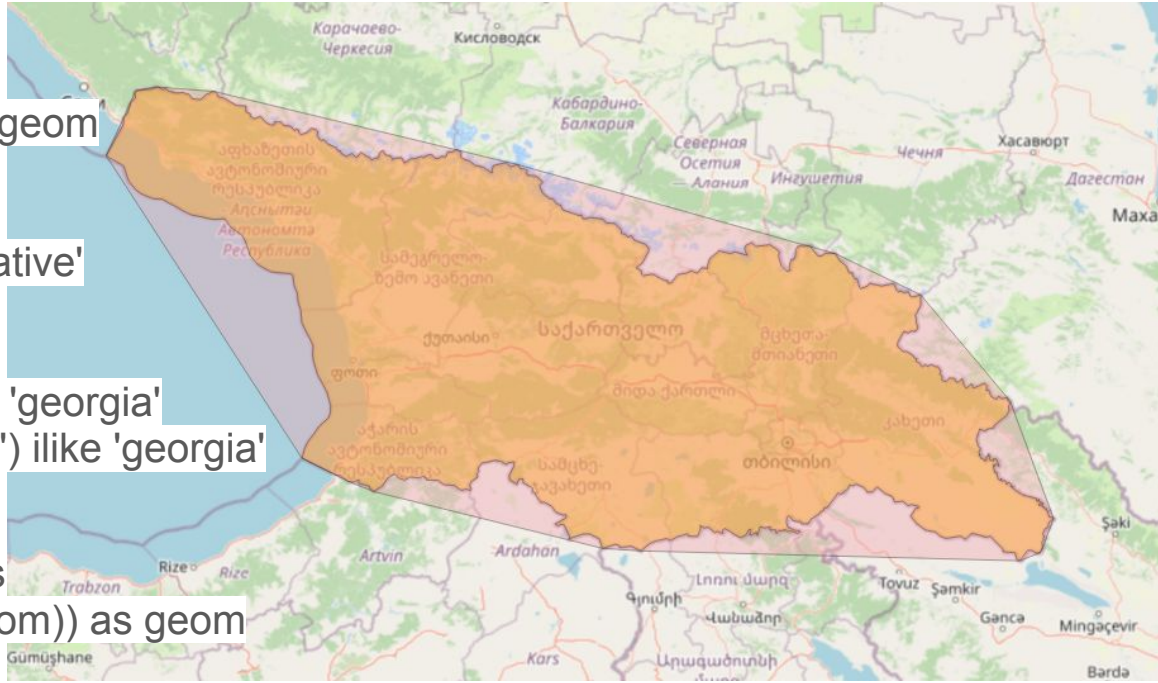
OpenStreetMap osmium import one-liner

```
osmium export -i sparse_mem_array -c osmium.config.json -f
pg data/in/osm/georgia-latest.osm.pbf -v --progress
| psql -1 -c "create table osm_georgia(geog geography,
osm_type text, osm_id bigint, version int, osm_user text, ts
timestamptz, way_nodes bigint[], tags jsonb); alter table
osm_georgia alter geog set storage external, alter osm_type
set storage main, alter osm_user set storage main, alter
way_nodes set storage external, alter tags set storage
external, set (fillfactor=100); copy osm_georgia from stdin
freeze;"
```

OpenStreetMap: Boundaries of Georgia

```
create table georgia_boundary as
select ST_Union(geog::geometry) as geom
from osm_georgia
where tags ? 'boundary'
  and tags ->> 'boundary' = 'administrative'
  and tags ->> 'admin_level' = '2'
  and (
        coalesce(tags ->> 'name', '') ilike 'georgia'
        or coalesce(tags ->> 'name:en', '') ilike 'georgia'
    );

create table georgia_convex_hull as
select ST_ConvexHull(ST_Collect(geom)) as geom
from georgia_boundary;
```



GEBCO: Global Bathymetric Chart of the Oceans

British Oceanographic Data Centre
very kindly provides elevation dataset that has both depth of the oceans and elevation above sea level.

15 arc-second interval grid.

111.3 km * 0.0041667 ≈ 463.8 m.

It nicely matches H3 Resolution 8.

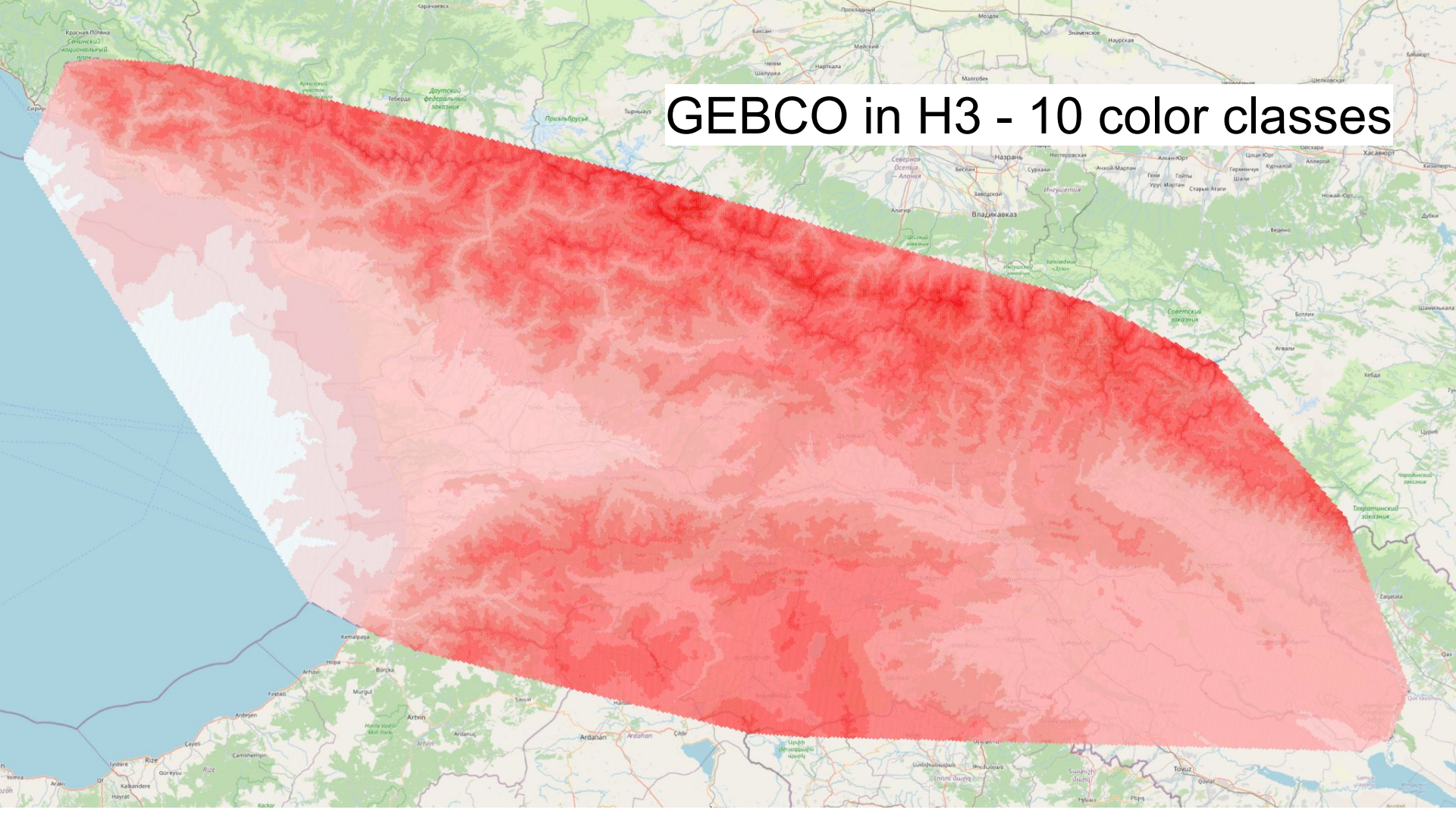
Importing GEBCO

```
$ raster2pgsql -I -C -M -s 4326 -t auto data/mid/gebco/*.tif
gebco_elevation_all | psql -q
```

```
$ psql -v table_name=gebco_elevation \
       -v table_name_h3=gebco_elevation_h3_r8 \
       -v item_name=ele \
       -v aggr_func=avg \
       -v resolution=8 \
       -v clip_table=georgia_convex_hull \
       -f scripts/raster_values_into_h3.sql
```

raster_values_into_h3.sql

```
-- Converts raster pixels to H3 aggregates using provided parameters
-- Expect psql variables: table_name, table_name_h3, item_name, aggr_func, resolution, clip_table
drop table if exists :table_name_h3;
create table :table_name_h3 as
with clip as (
    select geom from :clip_table
)
select
    H3,
    :resolution::int as resolution,
    agg_val as :item_name
from (
    select
        h3_latlng_to_cell(p.geom::geography, :resolution) as h3,
        :aggr_func(p.val) as agg_val
    from :table_name t,
         Clip,
         LATERAL ST_PixelAsCentroids(ST_Clip(t.rast, clip.geom, true)) as p
    group by 1
) s;
```

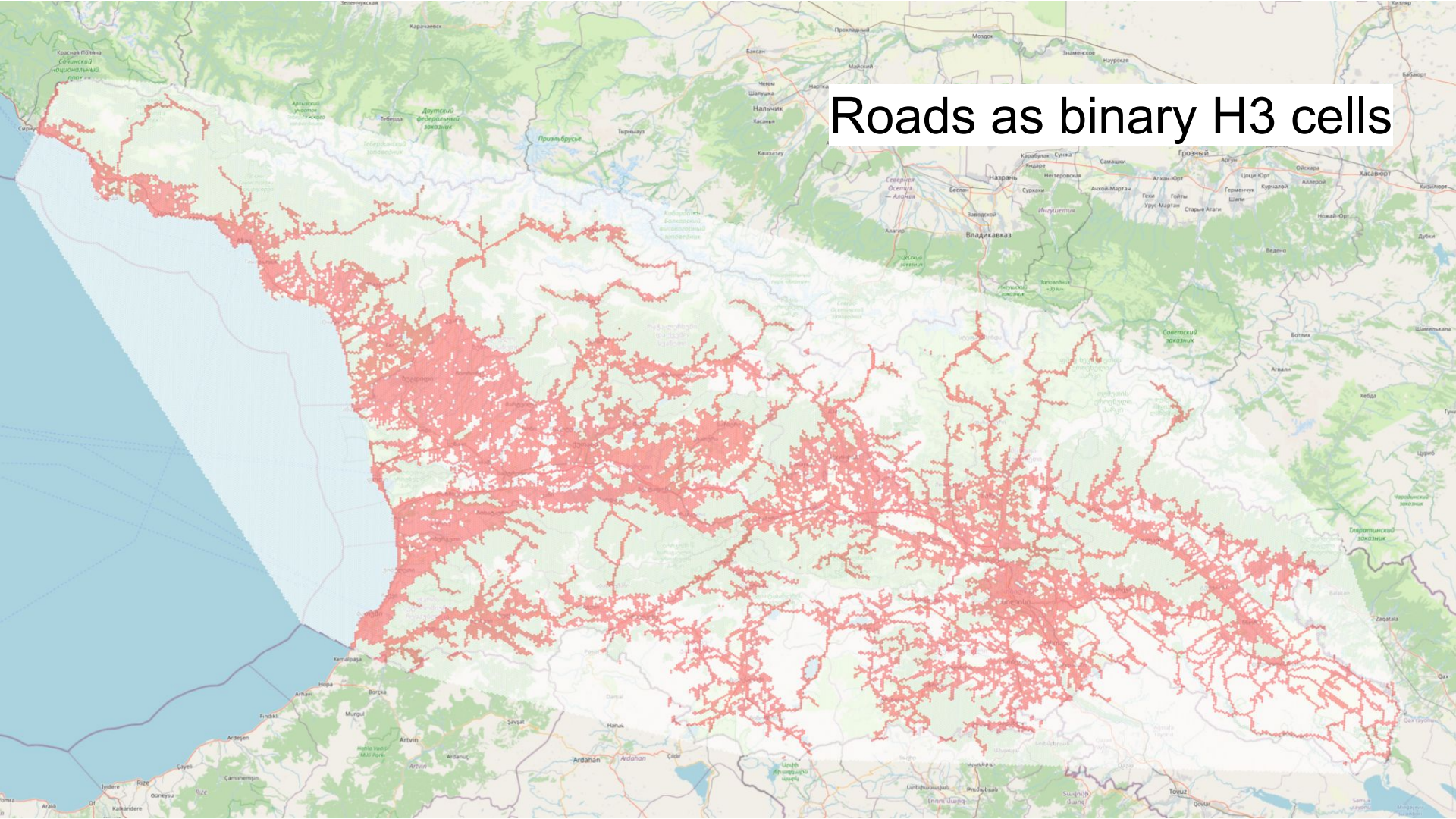

GEBCO in H3 - 10 color classes

Roads for motor vehicles

```
create table georgia_roads_geom as
select  geog::geometry as geom,
       tags ->> 'highway' as highway
from osm_georgia
where tags ? 'highway'
  and (tags ->> 'highway') = any (
        Array[
            'motorway', 'motorway_link', 'trunk', 'trunk_link',
            'primary', 'primary_link', 'secondary', 'secondary_link', 'tertiary', 'tertiary_link',
            'unclassified', 'residential', 'living_street', 'service', 'road'
        ] )
  and coalesce(
        lower(tags ->> 'motor_vehicle'),
        lower(tags ->> 'motorcar'),
        lower(tags ->> 'vehicle'),
        lower(tags ->> 'access'),
        'yes'
    ) not in ('no', 'private')
  and ST_GeometryType(geog::geometry) in ('ST_LineString', 'ST_MultiLineString');
```

Roads to Segments to Points to H3

```
create table roads_h3_r8 as
select
    h3_latlng_to_cell(ST_StartPoint(seg_geom)::geography, 8) as h3,
    sum(ST_Length(seg_geom::geography)) as road_length_m
from (
    select (ST_DumpSegments(
              ST_Segmentize(
                 geom::geography,
                 200)::geometry)
            ).geom as seg_geom
    from georgia_roads_geom
) segments
where seg_geom is not null
group by 1;
```


Roads as binary H3 cells

Population data in H3

```
$ ogr2ogr -f PostgreSQL PG:""
data/mid/population/kontur_population_20231101.gpkg -nln
kontur_population -nlt MULTIPOLYGON -lco GEOMETRY_NAME=geom
-overwrite -t_srs EPSG:4326
```

It’s already in H3.

Combine everything!

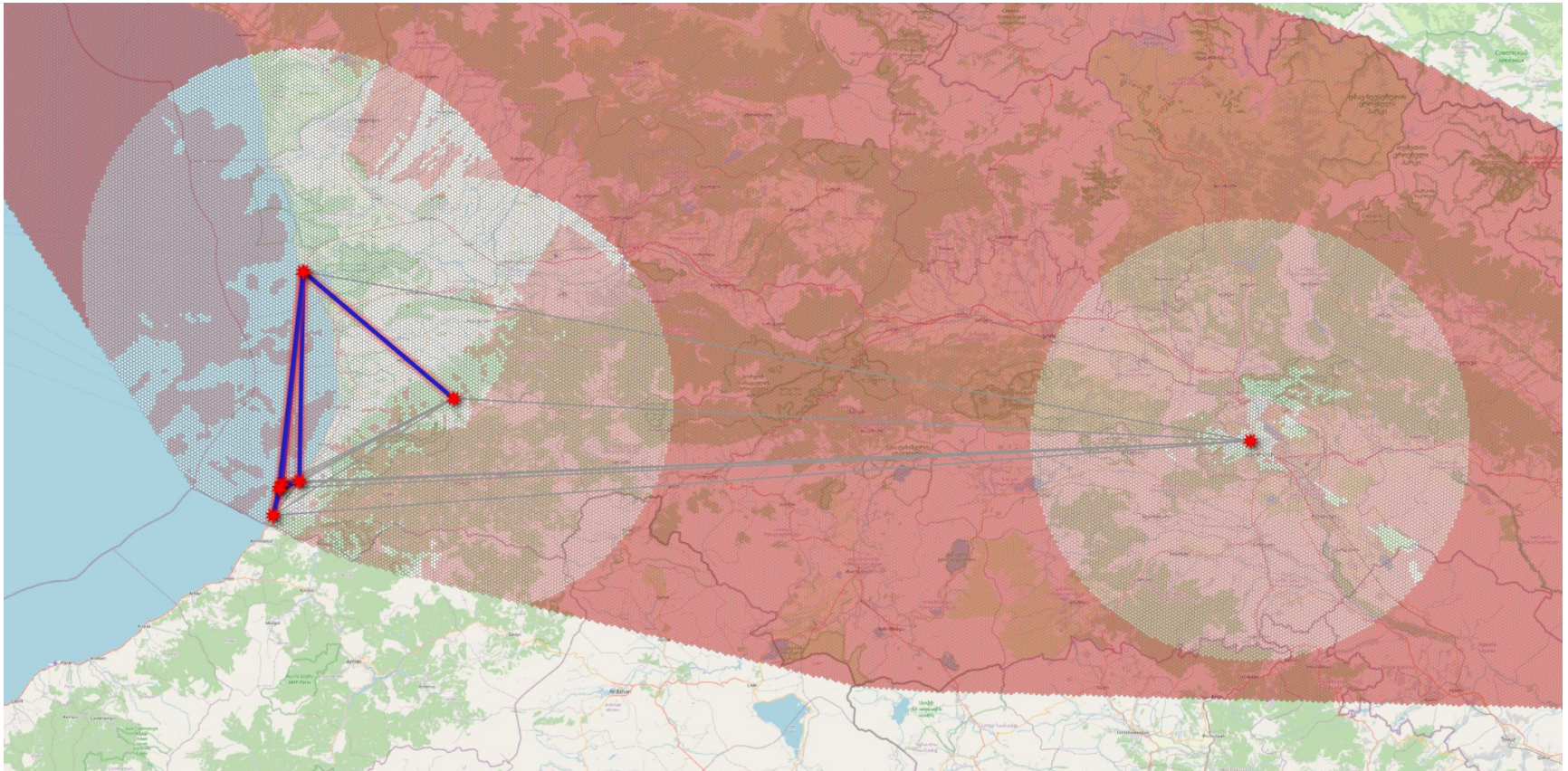
```
create table mesh_surface_h3_r8 (
    h3                              h3index primary key,
    geom geometry generated always as (h3_cell_to_boundary_geometry(h3))
stored,
    centroid_geog public.geography generated always as
(h3_cell_to_geometry(h3)::public.geography) stored,
    ele double                      precision,
    has_road                        boolean default false,
    population                      numeric,
    has_tower                       boolean default false,
    has_reception                   boolean,
    can_place_tower                 boolean,
    visible_uncovered_population    numeric,
    distance_to_closest_tower       double precision
);
```

Line of Sight Visibility function

```
create or replace function h3_los_between_cells(h3_a h3index,
h3_b h3index)
    returns boolean
    language plpgsql
    cost 100000
as
$$
…..

        /*  1) h3_grid_path_cells(norm_src, norm_dst) gives us
the H3 cells along
         *    the path between the towers (inclusive).
         * 2) We join that to mesh_surface_h3_r8 to get
elevations.
         * 3) For each intermediate cell:
         *      frac = position of cell centroid along the line
[0..1]
         *      los_height(frac) = ele_src + (ele_dst - ele_src)
* frac
         * If terrain elevation >= los_height at any point, LOS
is blocked.
         */
        with path as (
            select h3_grid_path_cells(norm_src, norm_dst) as h3
        ),
        samples as (
            select
                c.h3,
                ST_LineLocatePoint(
                    line_geom,
                    ST_PointOnSurface(c.geom)
                )::double precision as frac,
                c.ele
            from path p
            join mesh_surface_h3_r8 c
              on c.h3 = p.h3
            where p.h3 not in (norm_src, norm_dst)
        )
        select exists (
            select 1
            from samples s
            where s.frac > 0
              and s.frac < 1
              and (
                    ele_src + (ele_dst - ele_src) * s.frac
                  ) <= coalesce(s.ele, 0)
        )
        into has_blocker;
        is_visible := not coalesce(has_blocker, false);
    return is_visible;
```

Initial coverage: testing links



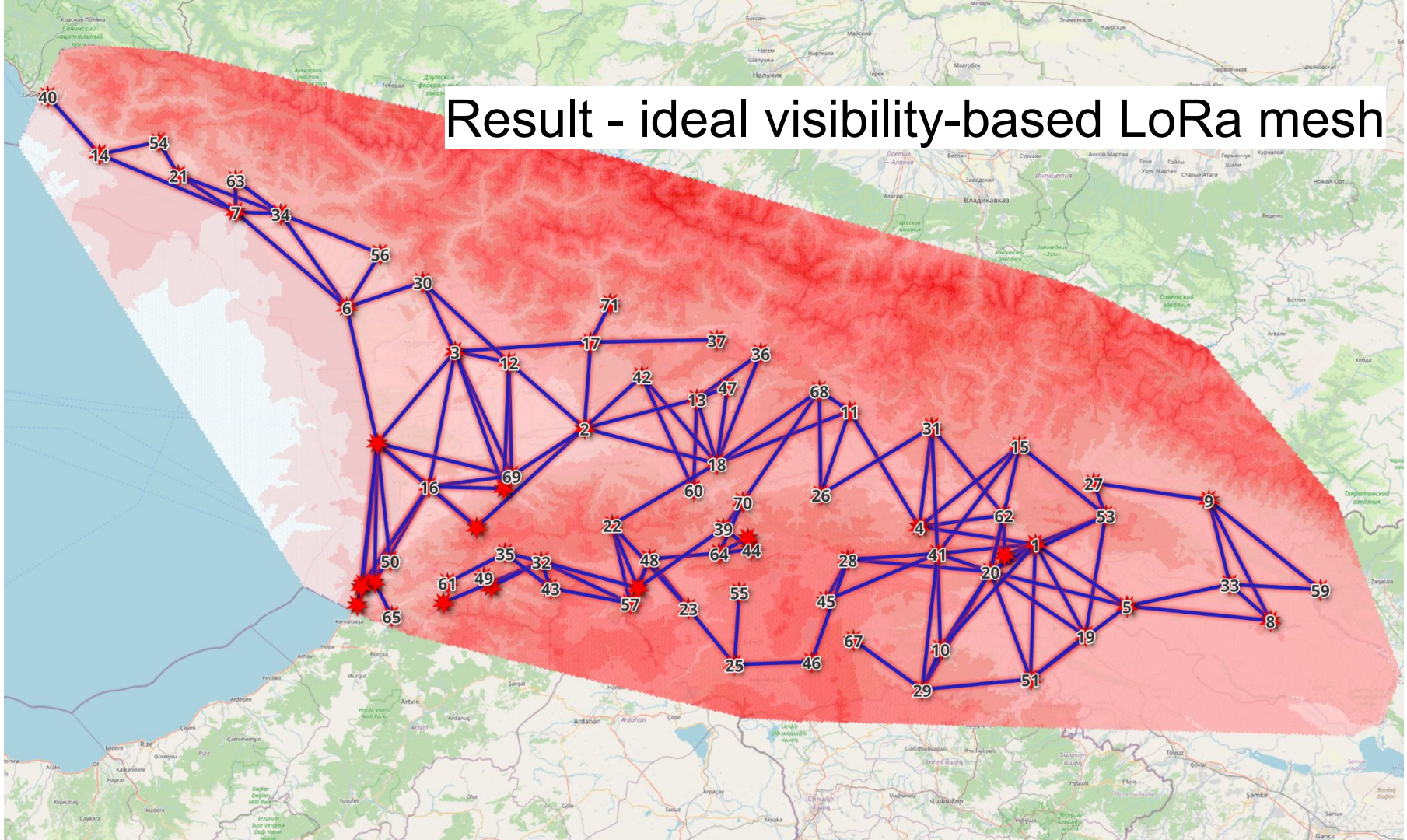
Greedy loop to pick growth direction

1. **Refresh reception cache**
   For each surface cell `s`:
   `has_reception(s) = exists tower t within 60 km with LOS`

2. **Enable tower candidates**
   `can_place_tower = true` where
   `has_road` and `not has_tower` and distance in `[5 km, 60 km]`.

3. **Score candidates by uncovered population**
   For each candidate `c`
   `visible_uncovered_population(c) = sum population(t)`
   over cells `t` that are
   `uncovered` + `within 60 km` + `LOS(c,t)`.

4. **Greedy pick + update**
   Choose `c*` with max `visible_uncovered_population`.
   Insert tower at `c*`, store iteration.
   Around `c*` (60 km): update `distance_to_closest_tower`, kill candidates closer than 5 km, invalidate caches.


Result - ideal visibility-based LoRa mesh
40
14
54
21
63
7
34
56
6
30
71
17
37
3
12
36
42
47
13
68
11
2
31
15
18
69
16
60
26
27
9
70
22
39
4
62
53
64
44
48
1
35
32
28
41
20
50
61
49
43
57
23
55
45
5
33
59
65
8
67
10
19
25
46
29
51

Useful links

https://github.com/Komzpa/h3-mesh-placement

https://github.com/konturio/geocint-runner/blob/main/DOCUMENTATION.md

https://meshtastic.org/

https://github.com/zachasme/h3-pg

https://www.gebco.net/

https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

Let’s be in touch

Darafei Praliaskouski
darafei@maumap.com