Looking into the past - feature extraction from historic maps using Python, OpenCV and PostGIS.

ESRC ADRC-S

- Administrative Data Research Centre – Scotland (ADRC-S)
- part of the Administrative Data Research Network (ADRN)
- An ESRC Data Investment
- 12 ADRC-S Work Packages
- EDINA working on WP5 - Provision of Geocoding and Georeferencing tools

What and Why?

Prof(s) Chris Dibben and Jamie Pearce from UoE GeoSciences

- Effects of past environmental conditions on (longitudinal) population cohorts

  Trains - where did they run in the past (causing air pollution from their coal fired boilers)

- Urban - did past populations live in predominantly urban or rural locales – have these same populations experienced become more urbanised.

- Industry - where were particular types of industry located

- Greenspace and Bluespace - Parks and Water

Historic Maps – a record of past landscapes

- ADRC`s remit is (all) of Scotland.
- Manual capture of features from historic maps not going to scale given resources available.
- Chris and Jamie`s challenge to EDINA – is it possible to automagically capture features from historic maps?
- Historic maps in Digimap historic
- For the purpose of this work we are using (higher quality) full colour scans of historic maps provided by Chris@NLS
- Mainly been looking at 2 map series provided by NLS

Image Processing & Computer Vision

Tasks that are easy for a human to carry out are much harder for computers.

Environment

- Ubuntu
- Python
- Virtualenv
- PyCharm (Community Edition) Python IDE
- OpenCV
- PostgreSQL
- PostGIS
- QGIS
- (a bit of) ArcGIS for ArcScan

Python Libraries used

- **numpy** - numpy array data structures sit at the heart of everything else
- **cv2** - python interface to OpenCV
- **shapely**
- **fiona**
- **rasterio**
- **Snaql** - "Raw (S)QL queries in Python without pain"

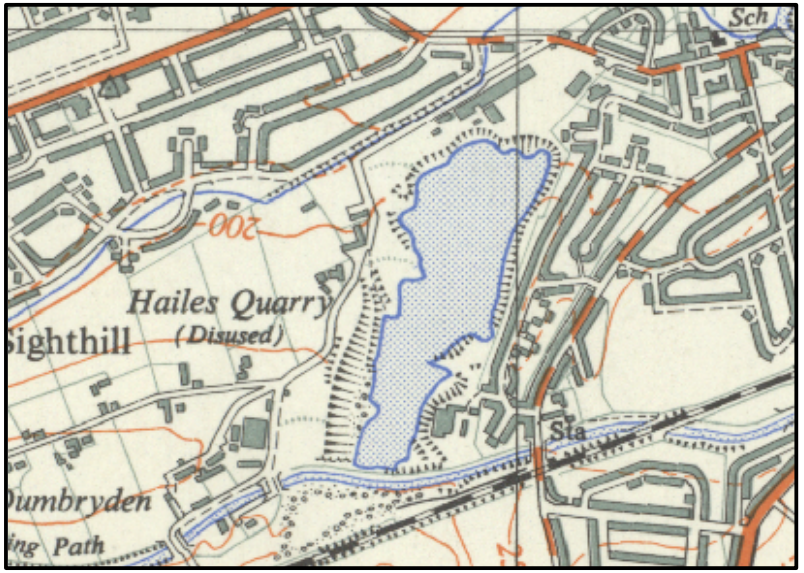
  assuming **PostGIS**, if you add in a map renderer like **mapnik**, then this lot gives you everything needed to do geospatial data analysis (raster and vector), data conversion, data management and map automation.

Python OpenCV Demo

- Load image
- Changing colourspaces
- Image segmentation / thresholding
- Finding image contours
- Finding patterns / classifying features

(1) Water features (Bluespace)

Rivers / Canals / inland water shown as blue lines or stippled blue areas.

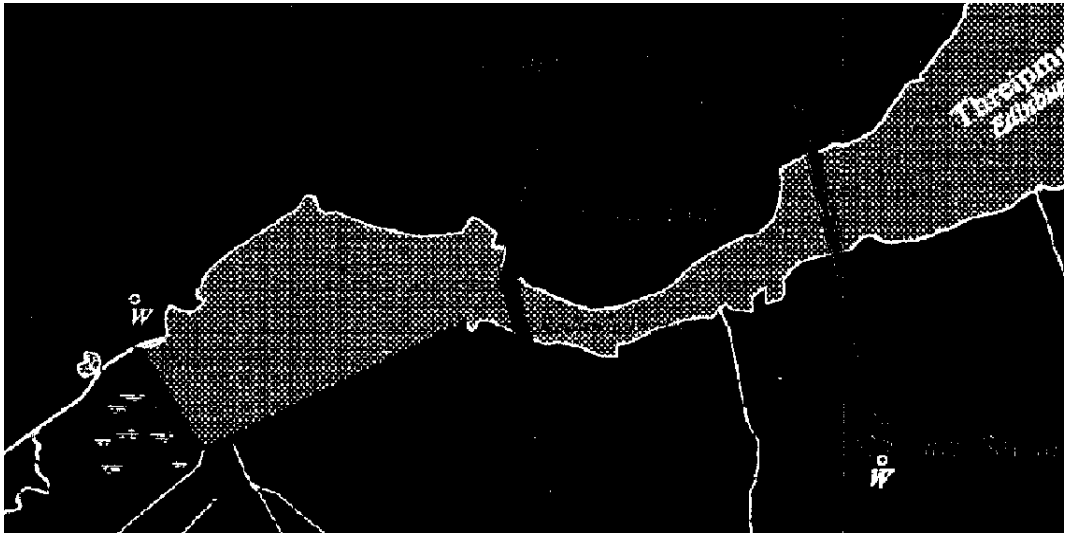


Threshold to isolate blue pixels

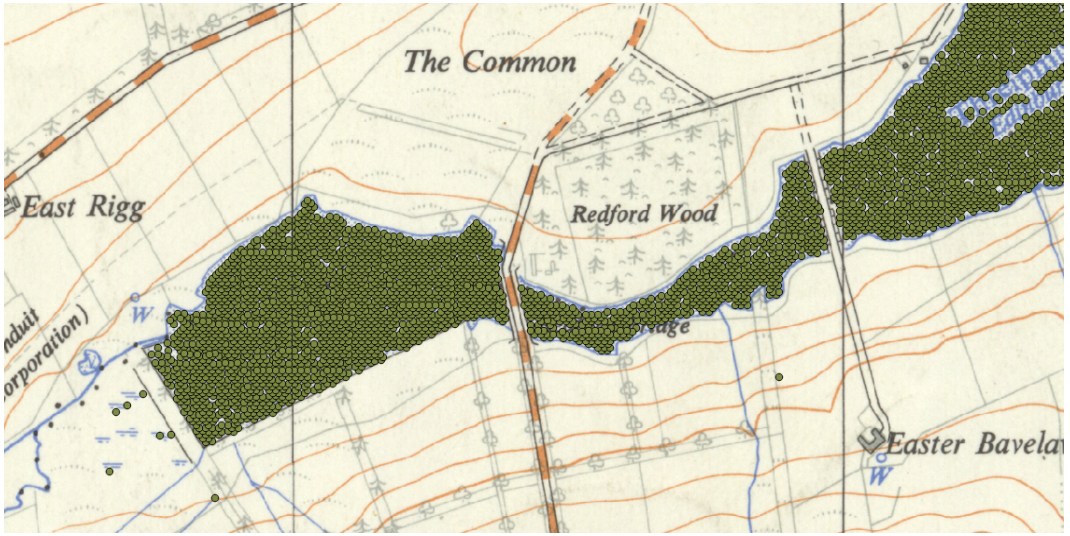
Find contours – each stipple mark / line forms a contour

Contours form a hierarchy. Parents that hold child contours are water regions.

Method 2

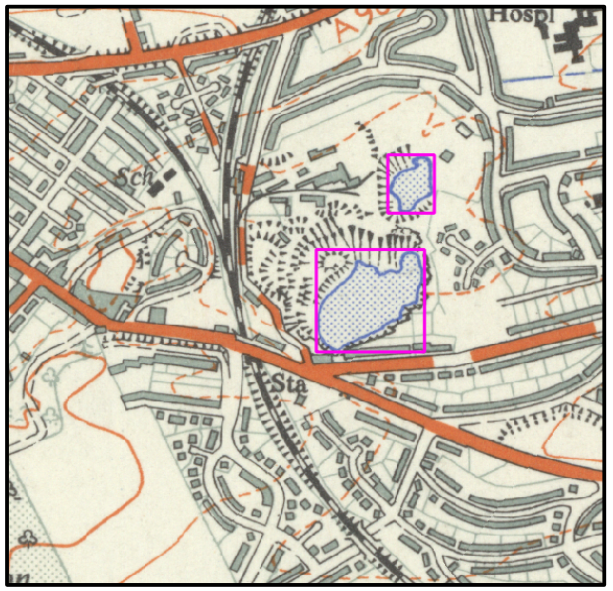


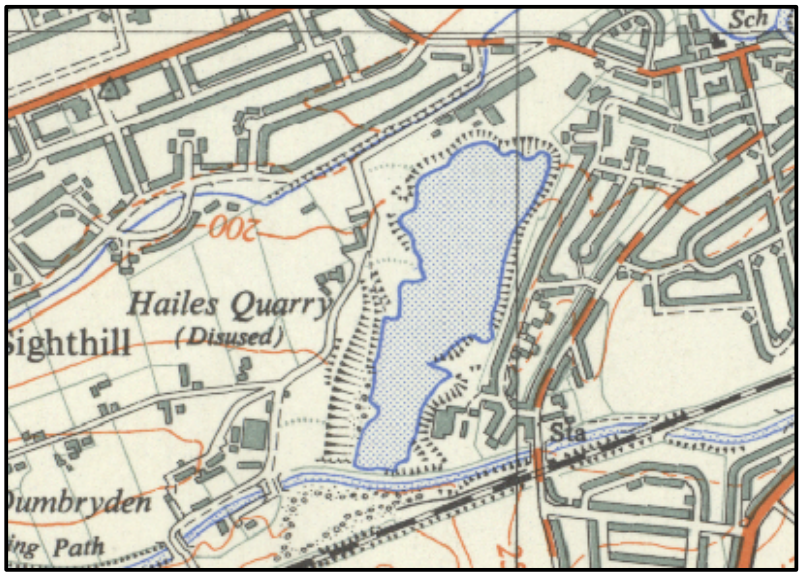
Process breaks down when water regions are not entirely bound by blue lines or broken by other features (bridges).



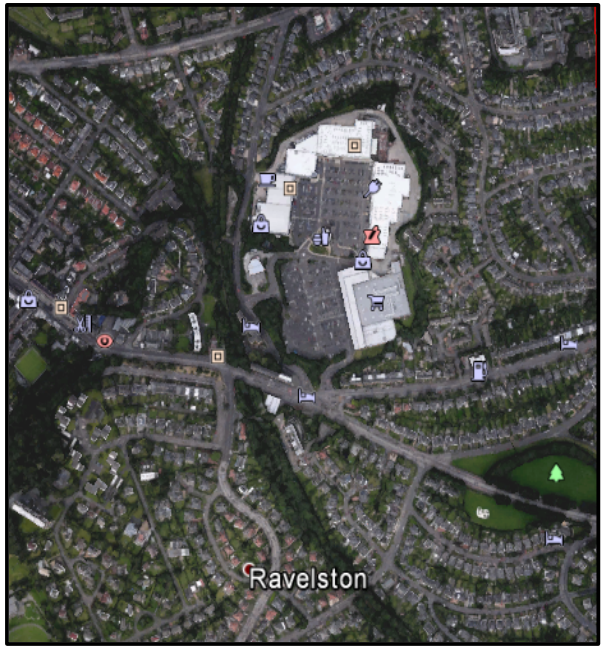
So (alternative method) find every individual stipple and then forming groups of these gives water regions.

Apply either of these methods of capturing blue stippled regions to other stippled regions e.g. green stippled regions (parks - greenspace)



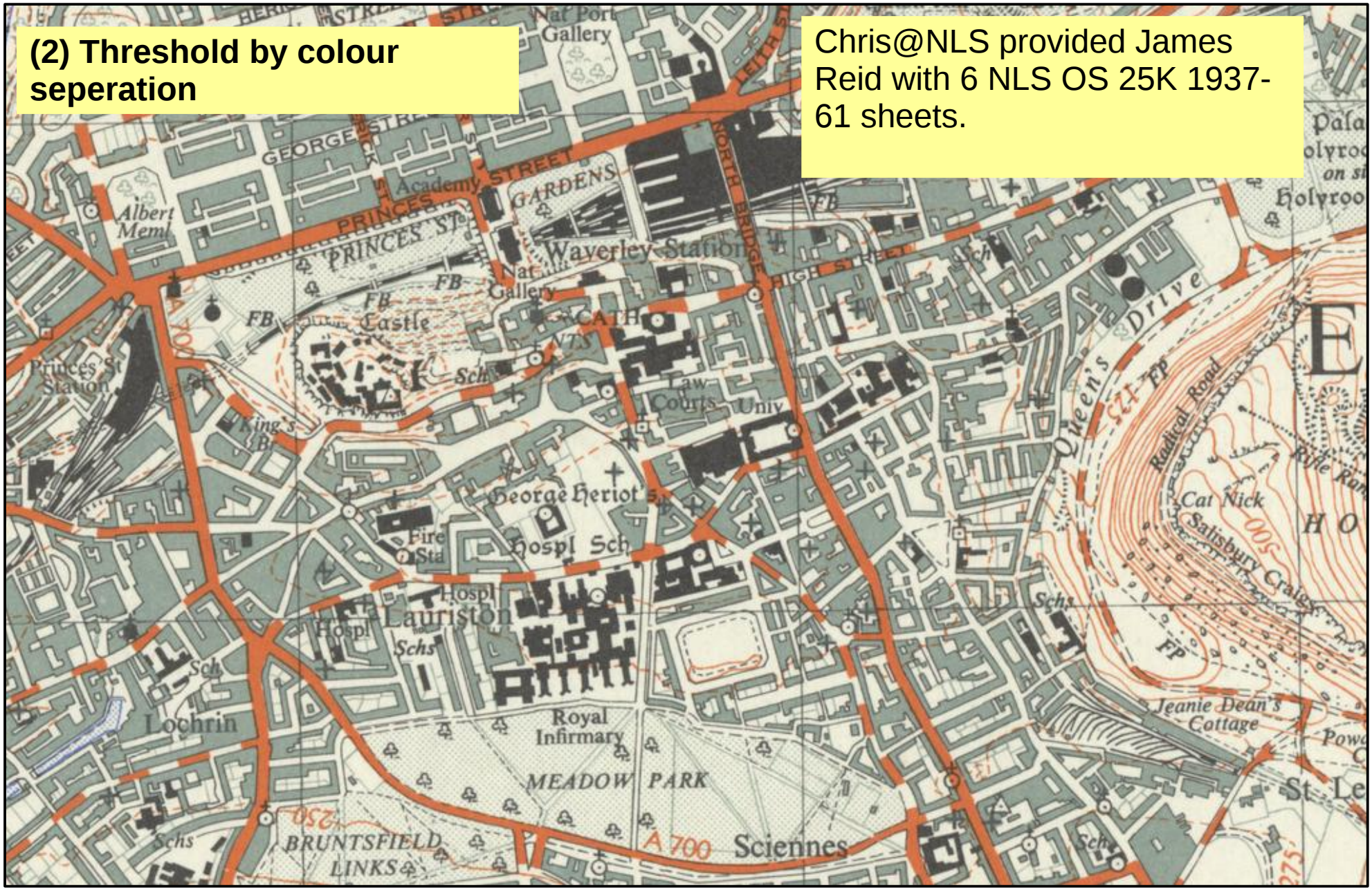


Change - old Edinburgh quarries change to shopping centres or from bluespace to greenspace!

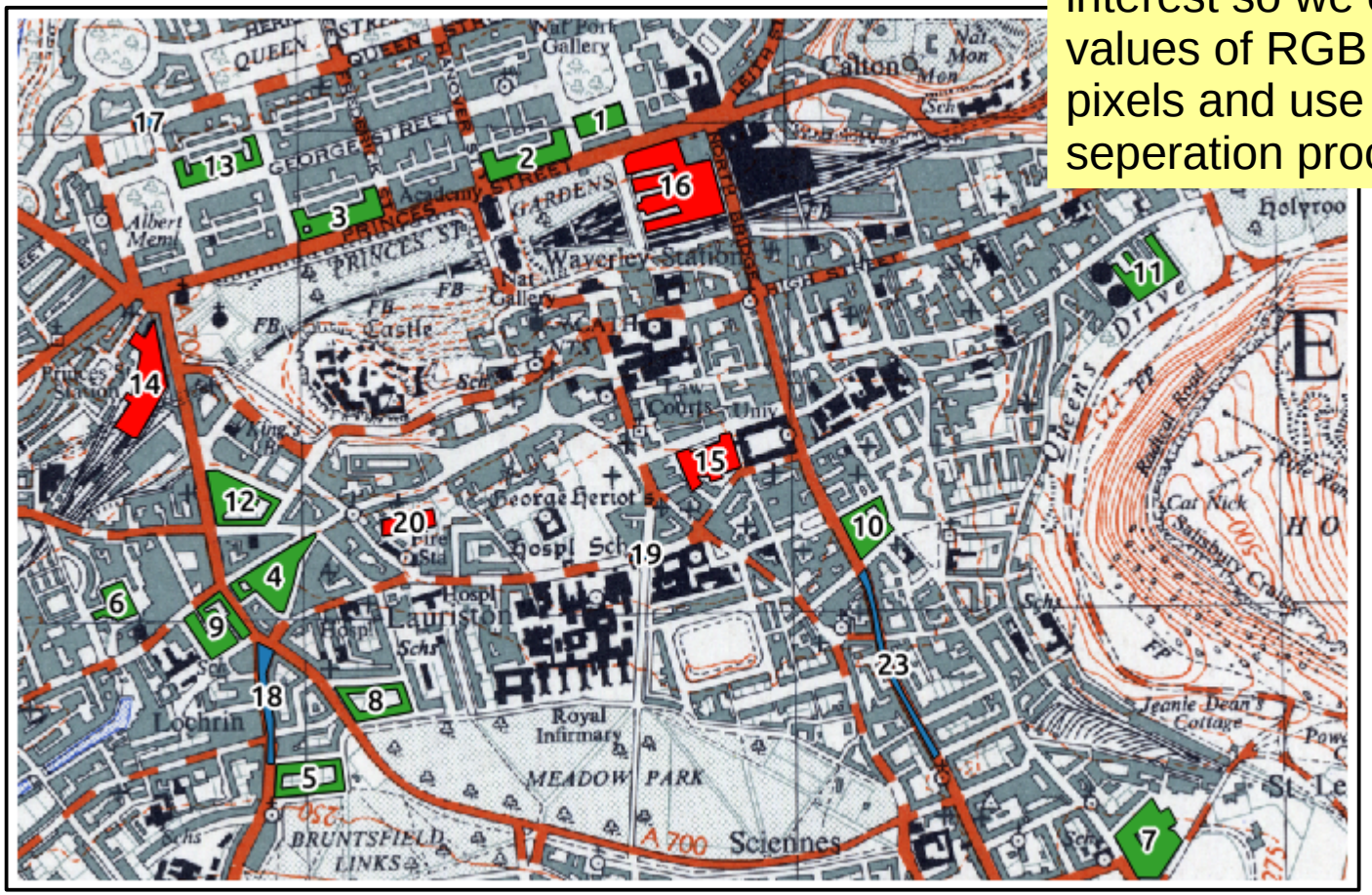


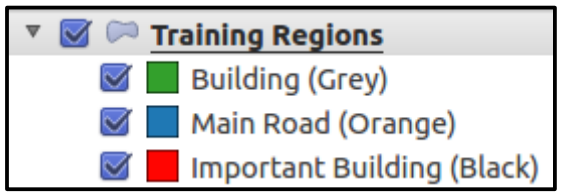
(2) Threshold by colour seperation

Chris@NLS provided James Reid with 6 NLS OS 25K 1937-61 sheets.

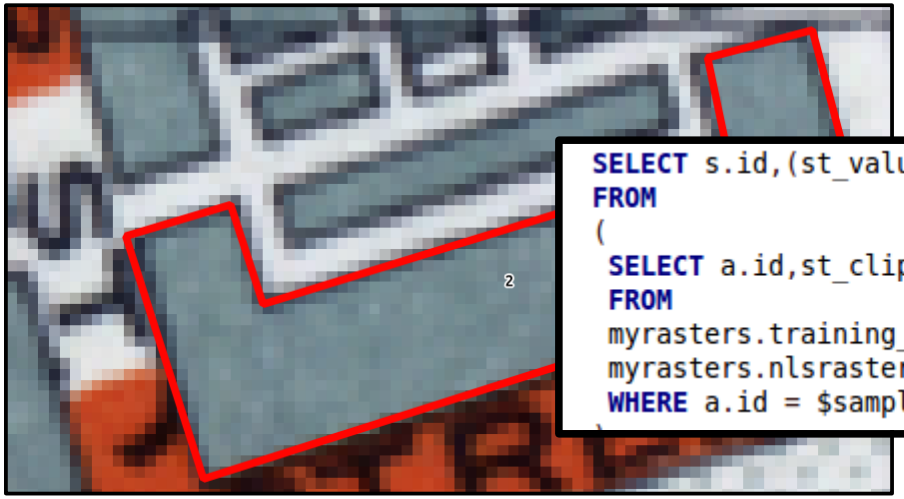


In QGIS digitised polygons covering groups of features of interest so we can explore values of RGB in the underlying pixels and use to inform colour seperation processing.





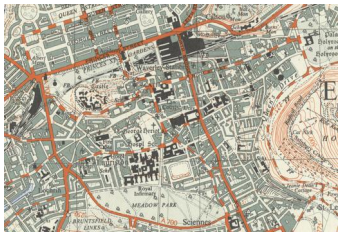
Load the training polygons and NLS 3 band raster into PostGIS and do spatial analysis to find pixel values in each polygon.

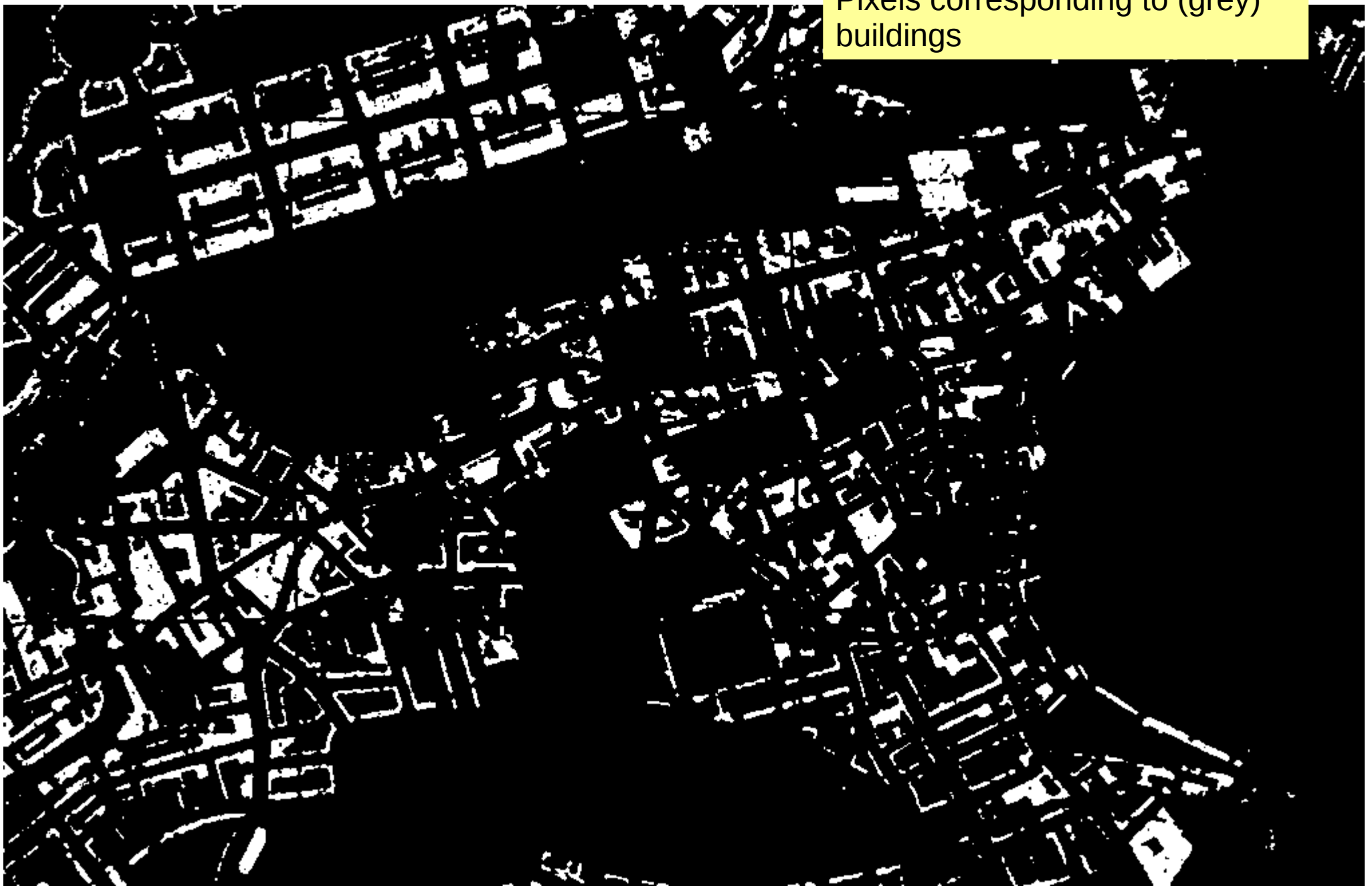


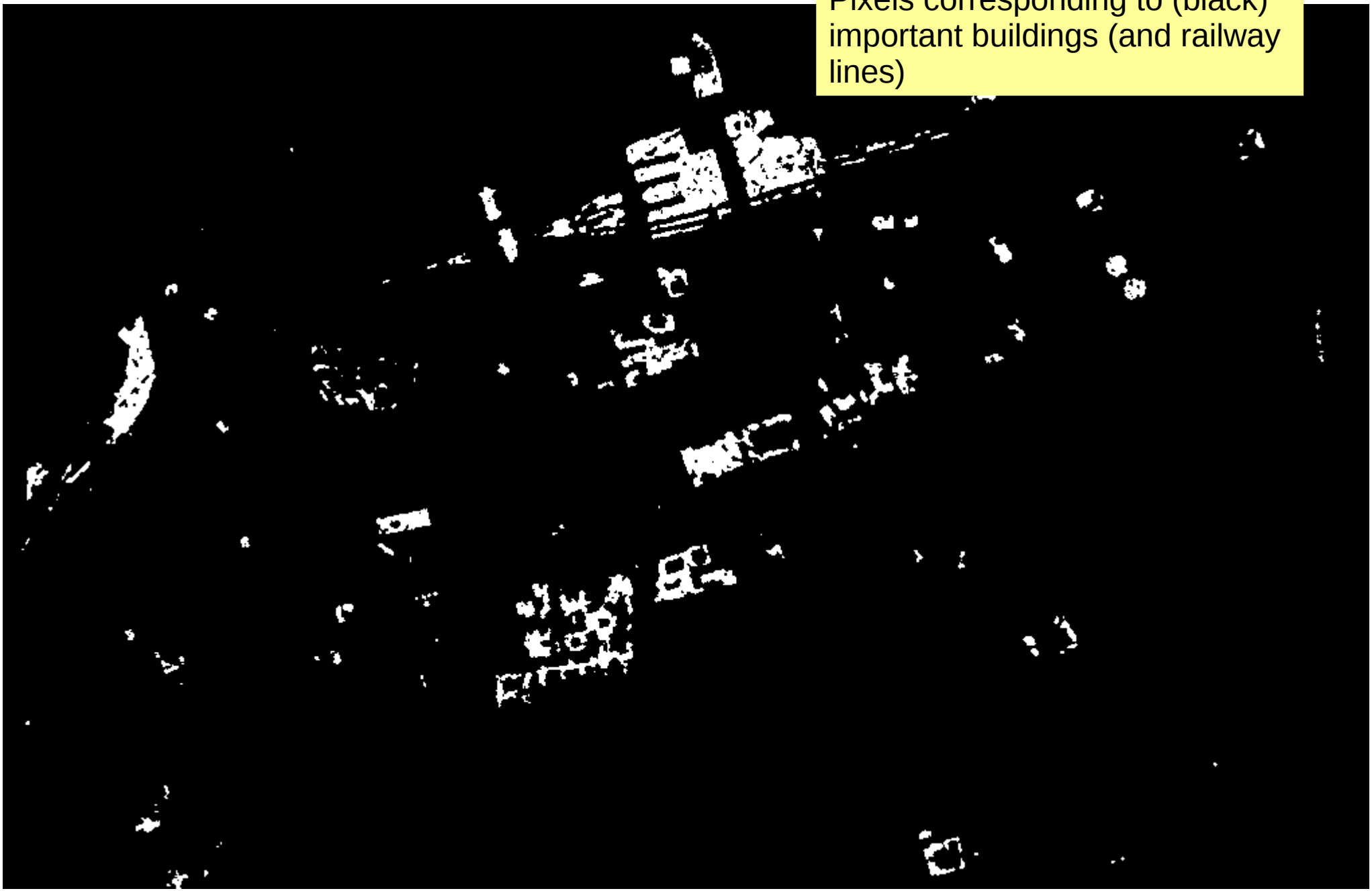
```
SELECT s.id,(st_valuecount(s.clipped_raster, $band_id,true)).*
FROM
(
 SELECT a.id,st_clip(b.rast, a.geom,true) as clipped_raster
 FROM
 myrasters.training_regions a,
 myrasters.nlsraster b
 WHERE a.id = $sample_region_id
```

sample_region	ftype	min_b	min_g	min_r	max_b	max_g	max_r
1	building	148	161	147	155	168	154
2	building	146	159	145	155	168	154
3	building	147	160	146	156	169	155
4	building	147	160	146	156	169	155
5	building	147	160	147	156	169	156
6	building	148	160	147	157	169	156
7	building	146	158	144	155	167	153
8	building	145	160	146	156	169	155
9	building	143	157	144	153	166	153
10	building	146	159	145	155	168	154
11	building	147	160	146	156	169	155
12	building	146	158	146	155	169	155
13	building	147	160	146	156	169	155
17	Road	67	111	210	76	120	219
18	Road	67	110	209	76	119	218
19	Road	66	108	208	75	118	217
21	Road	68	110	207	77	119	216
22	Road	70	111	207	79	120	216
23	Road	67	109	206	76	118	215
14	SBuilding	71	74	76	80	84	85
15	SBuilding	71	74	75	80	83	85
16	SBuilding	68	70	72	77	79	81
20	SBuilding	71	75	76	80	84	85

Calculate aggregate min/max values of RGB (BGR in opencv!) across each feature group and use these in OpenCV Python algorithm to do colour seperation on the source 25k image. More pre/post processing needed.



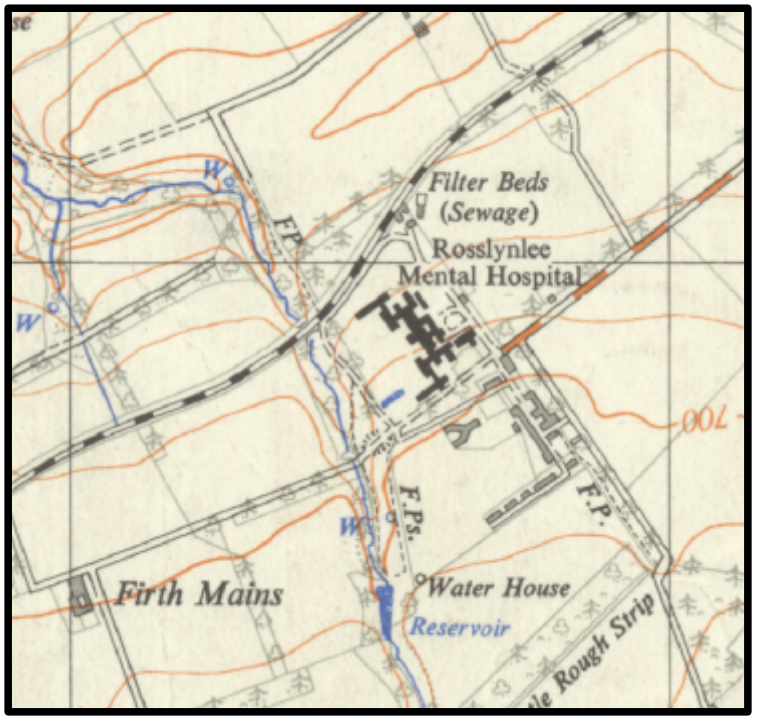

Pixels corresponding to (grey) buildings



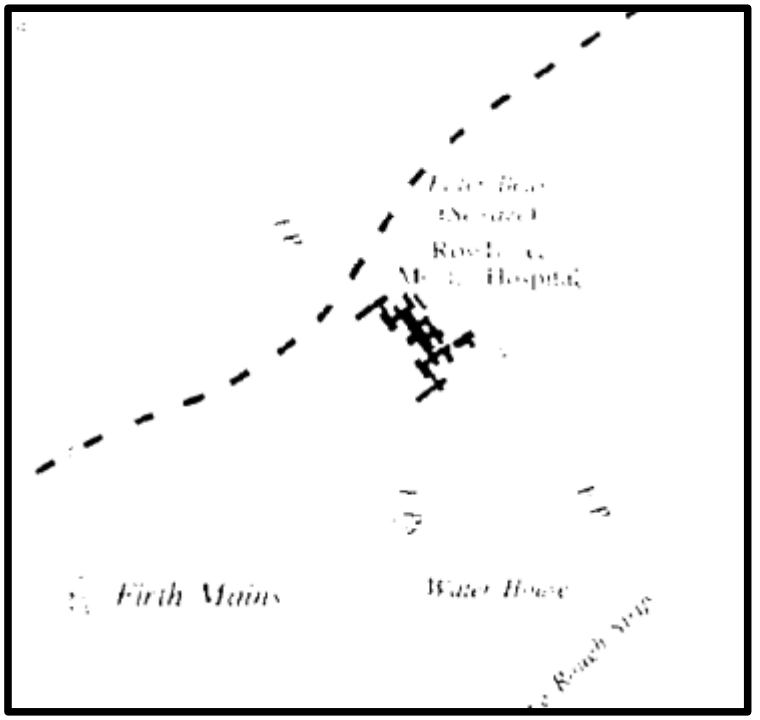
Pixels corresponding to (black) important buildings (and railway lines)


Pixels corresponding to
(orange) main roads

(3) Extracting Railways

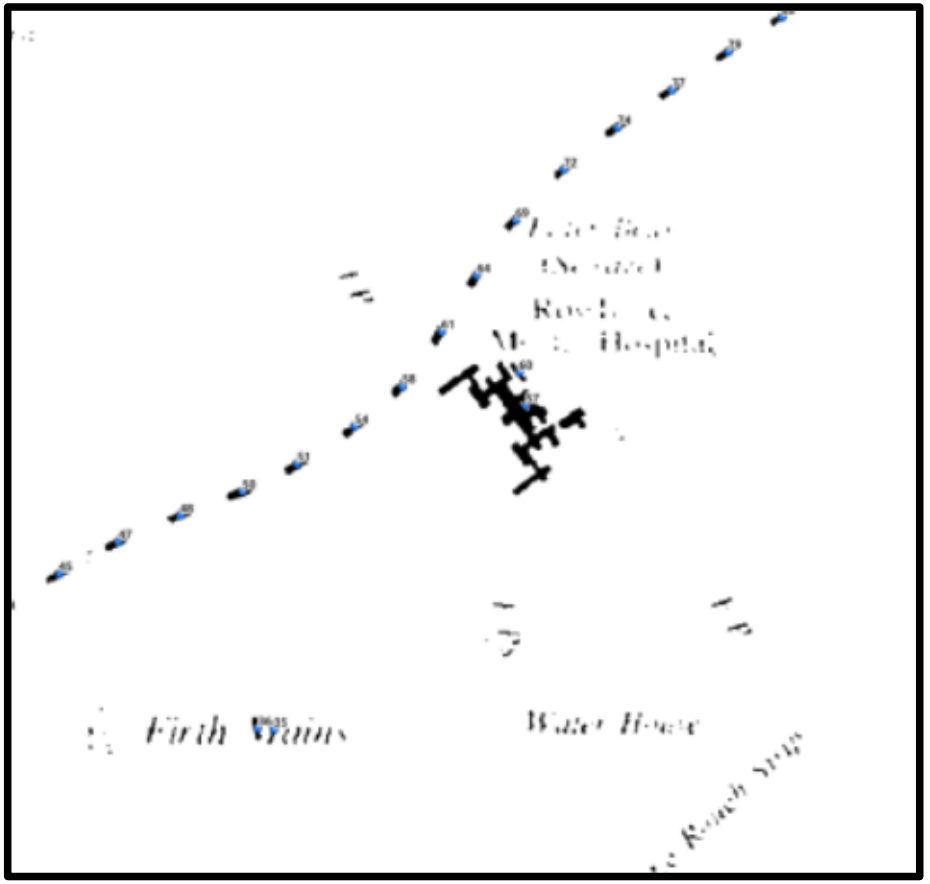


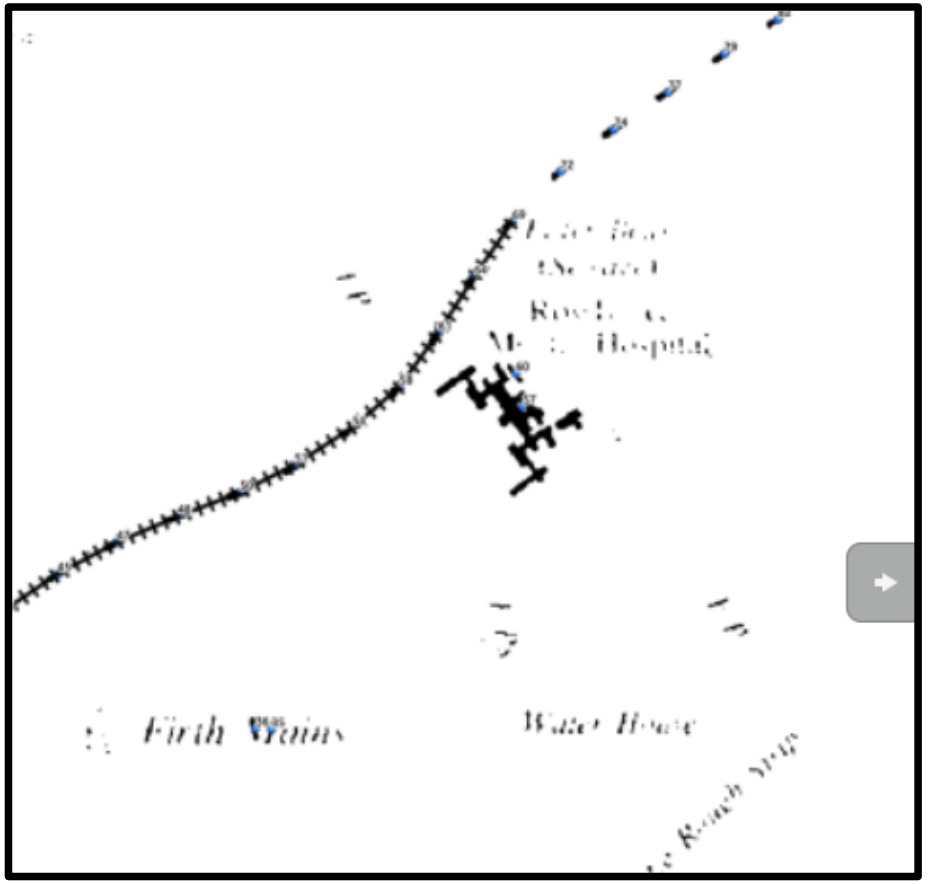
Source 1:25,000 NLS Historic Map



"black" pixels extracted after running colour seperation process

From dashes to (railway) lines





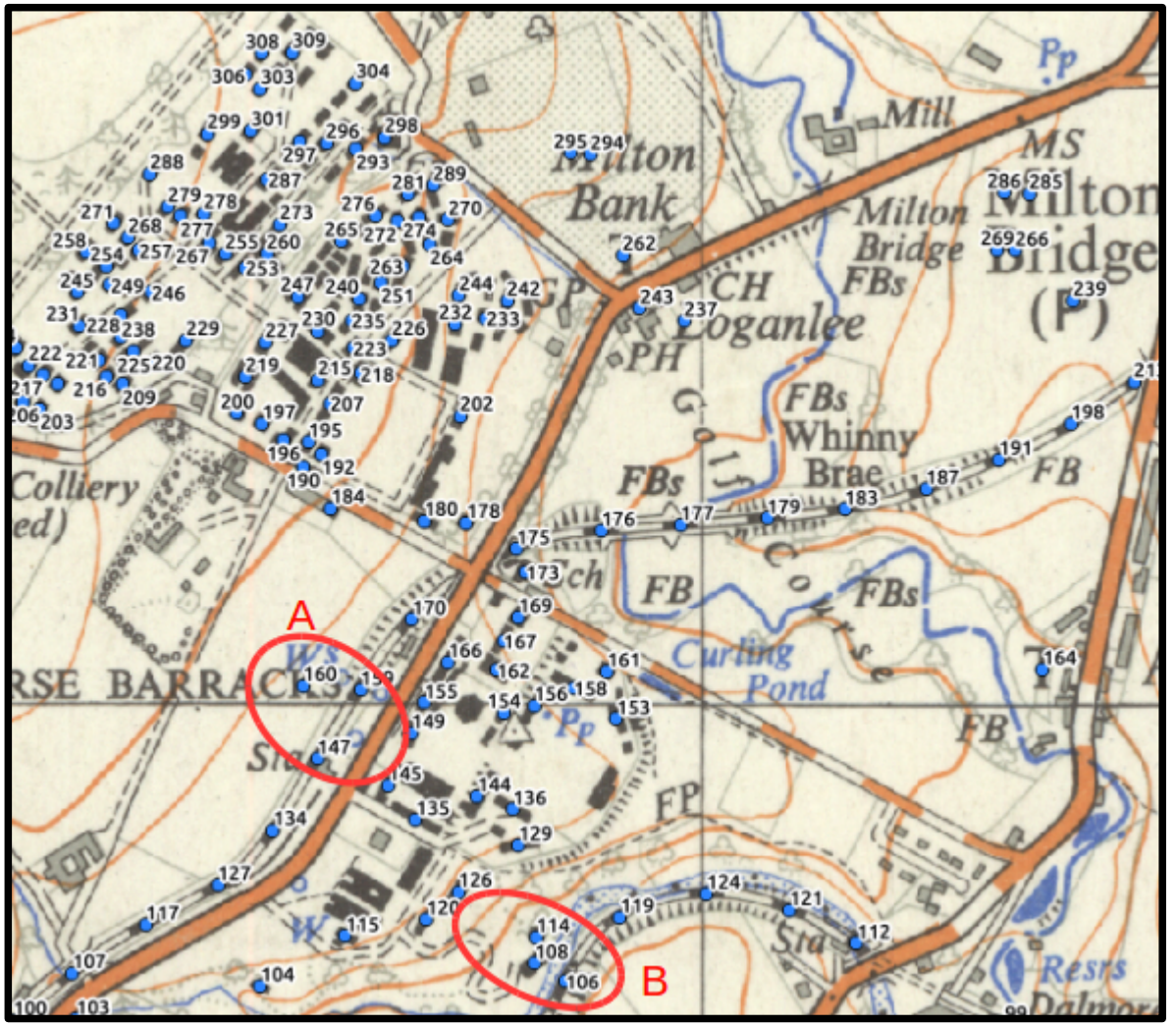
Marked up candidates (blue dot centroid) for dashes in railway lines After doing contour tracing and selecting contours that meet size constraints.

Join up neighbouring dash candidates to form railway line (polyline shapefile)

Complications…

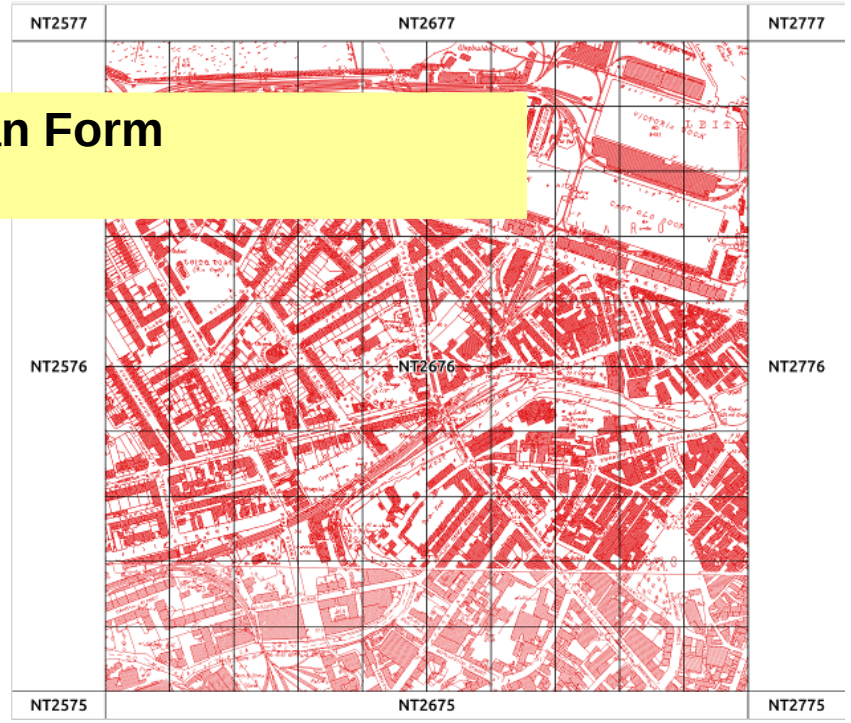
Process needs refined to cope with noisier, more complicated regions of the map.

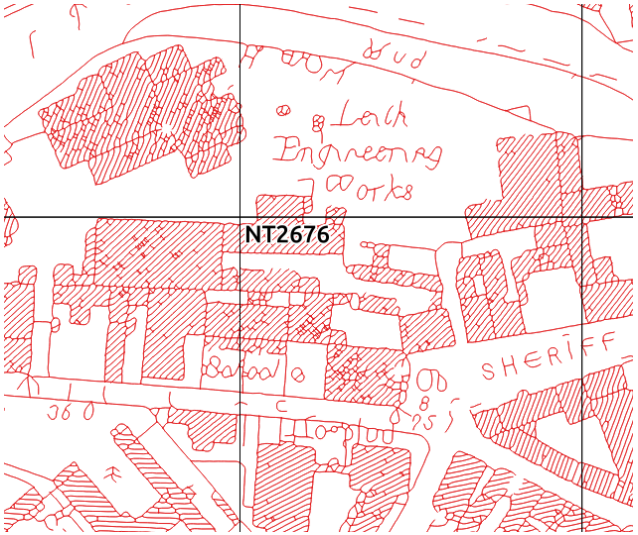
A refinement might be to introduce a look-ahead constraint that minimises change in line direction as candidates are grouped since railway lines don`t make sharp 90 degree turns.



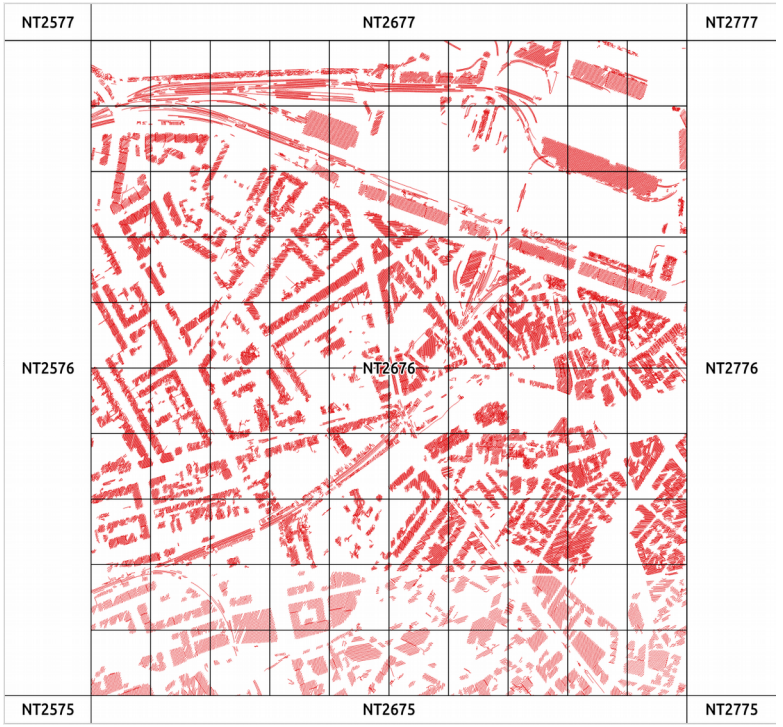
(4) Urban Form

All lines captured from different historic NLS ca1900 Map series





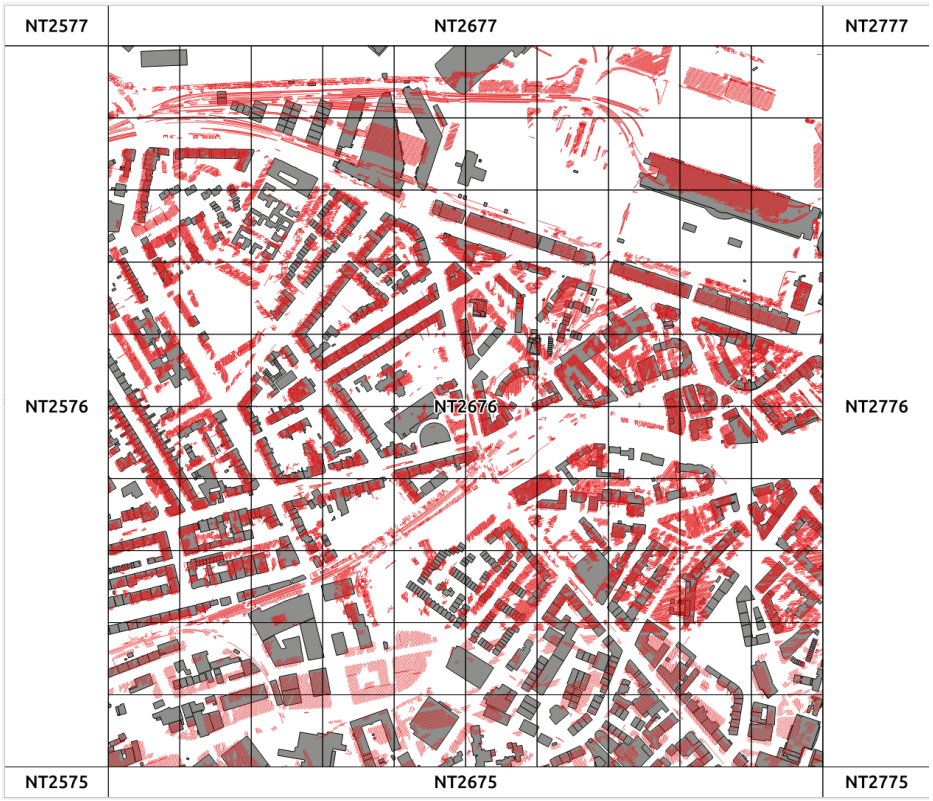
Spatial analysis



Left with lines corresponding to hatched building regions

New vs Old (Buildings)

Current building footprints held in OS MasterMap

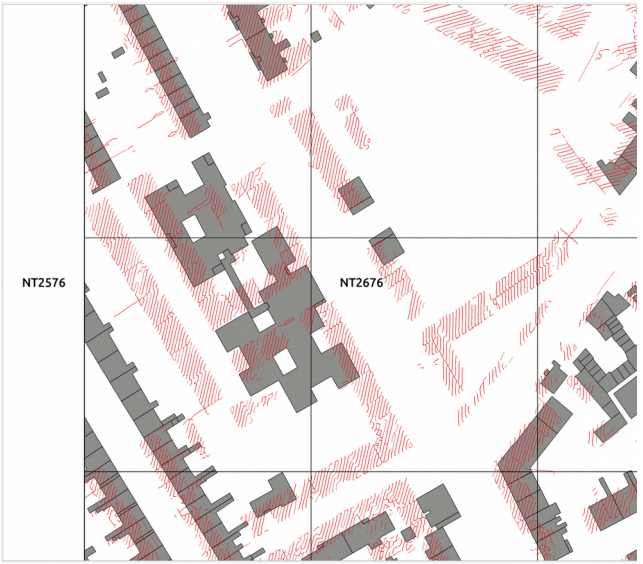


Lines from historic map selected as corresponding to hatched building areas overlain against OSMM building footprints

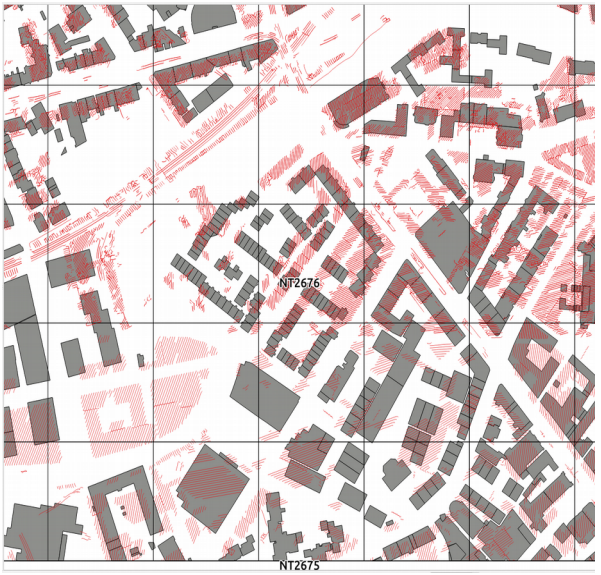
Examples of change in Edinburgh between ca1900 and today

All change

The locale of the Fort public housing project.



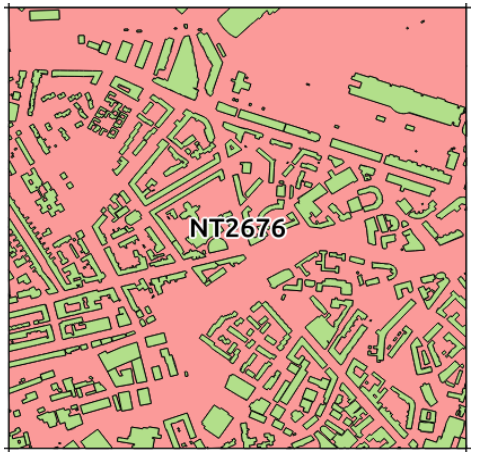
West Bowling Green Street & Bowling Green Street



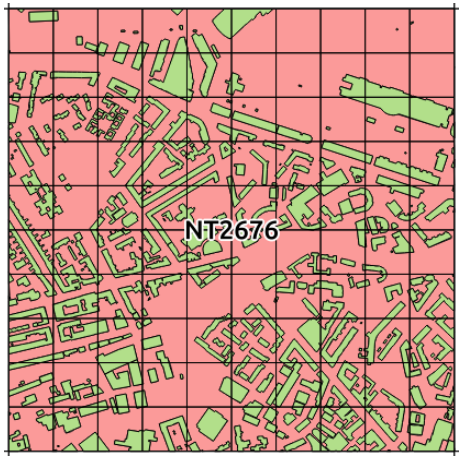
A measure of urbaness

Discrete building areas

Dissolve



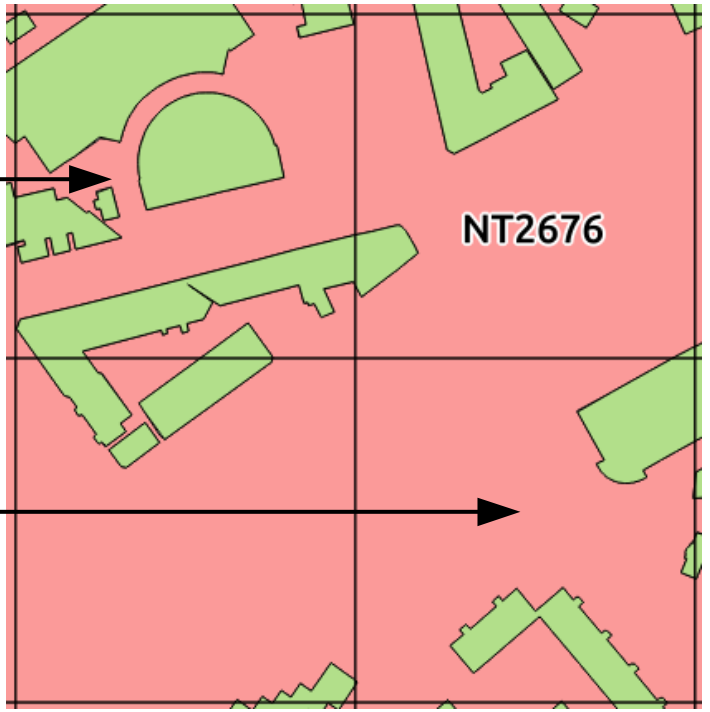
is_building = Yes / No



Overlay a 100m x 100m sampling grid

% Building = Higher

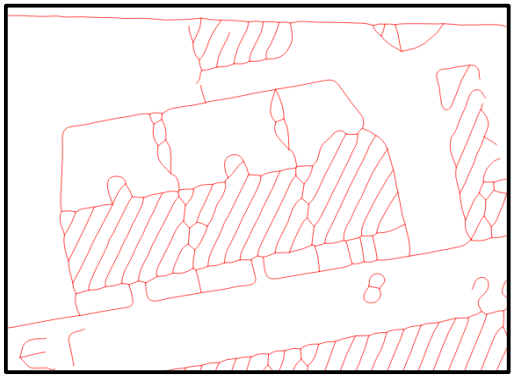
% Building = Lower



From hatch lines to buildings

1. All lines pulled by from NLS historic map sheet. No intelligence about what each line represents.

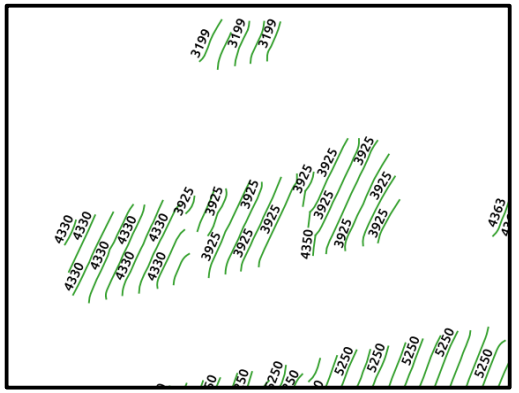
Spaghetti!



2. Form groups of hatch lines.

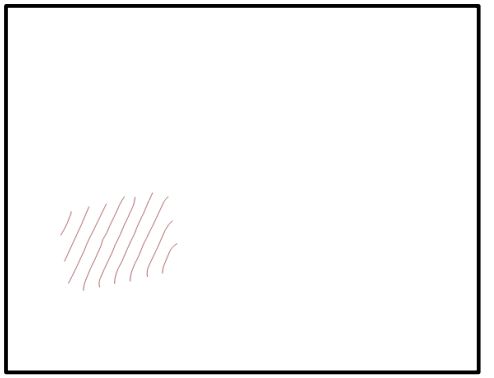
Criteria for group membership is: spatial proximity; direction (azimuth); lines are spatially disjoint; lines are parallel to one another.

3. Final set of line groups. These correspond to building footprint. Other lines from the historic map did not meet group membership criteria and thus make no further contribution to analysis.

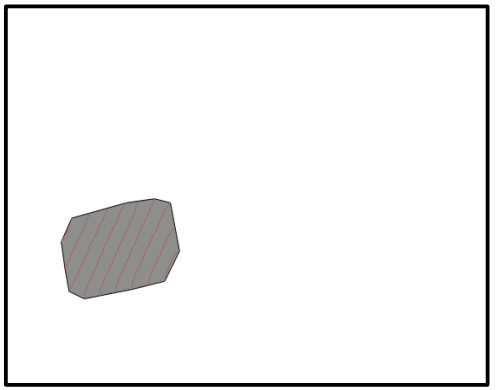


4. Derive a pseudo building polygon for each group.

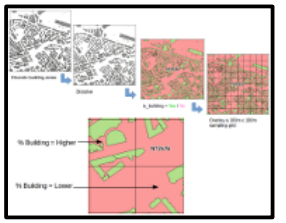
Could place an MBR around them but instead...



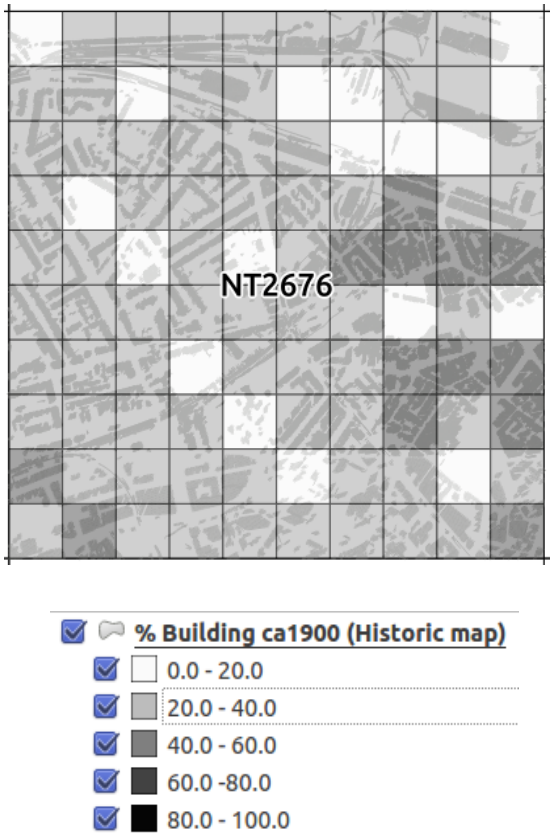
5. … form a **Convex Hull** around the lines to provide a polygon for this group. *For the historic maps this is the equivalent of the building footprint provided by the OS MasterMap data.*

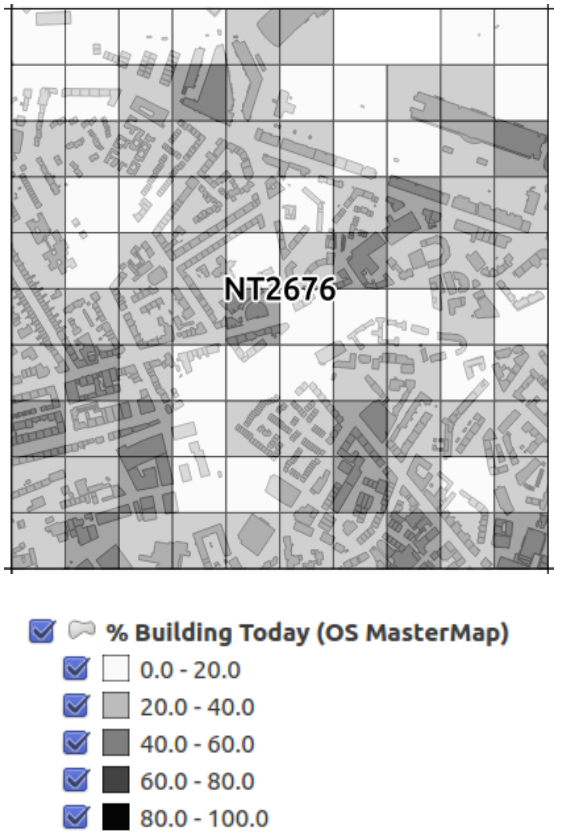


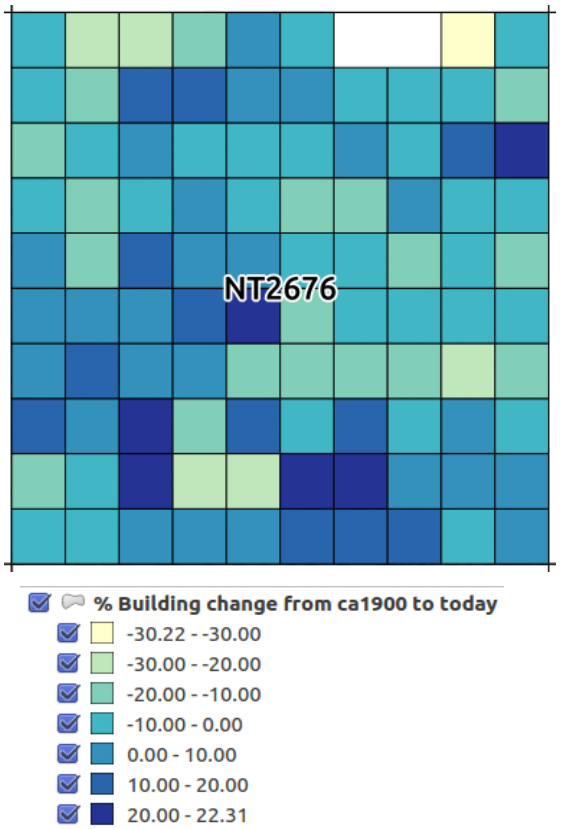
6. Repeat the % Building analysis for the complete set of convex hull polygons formed from all groups of hatch lines.



Output data products

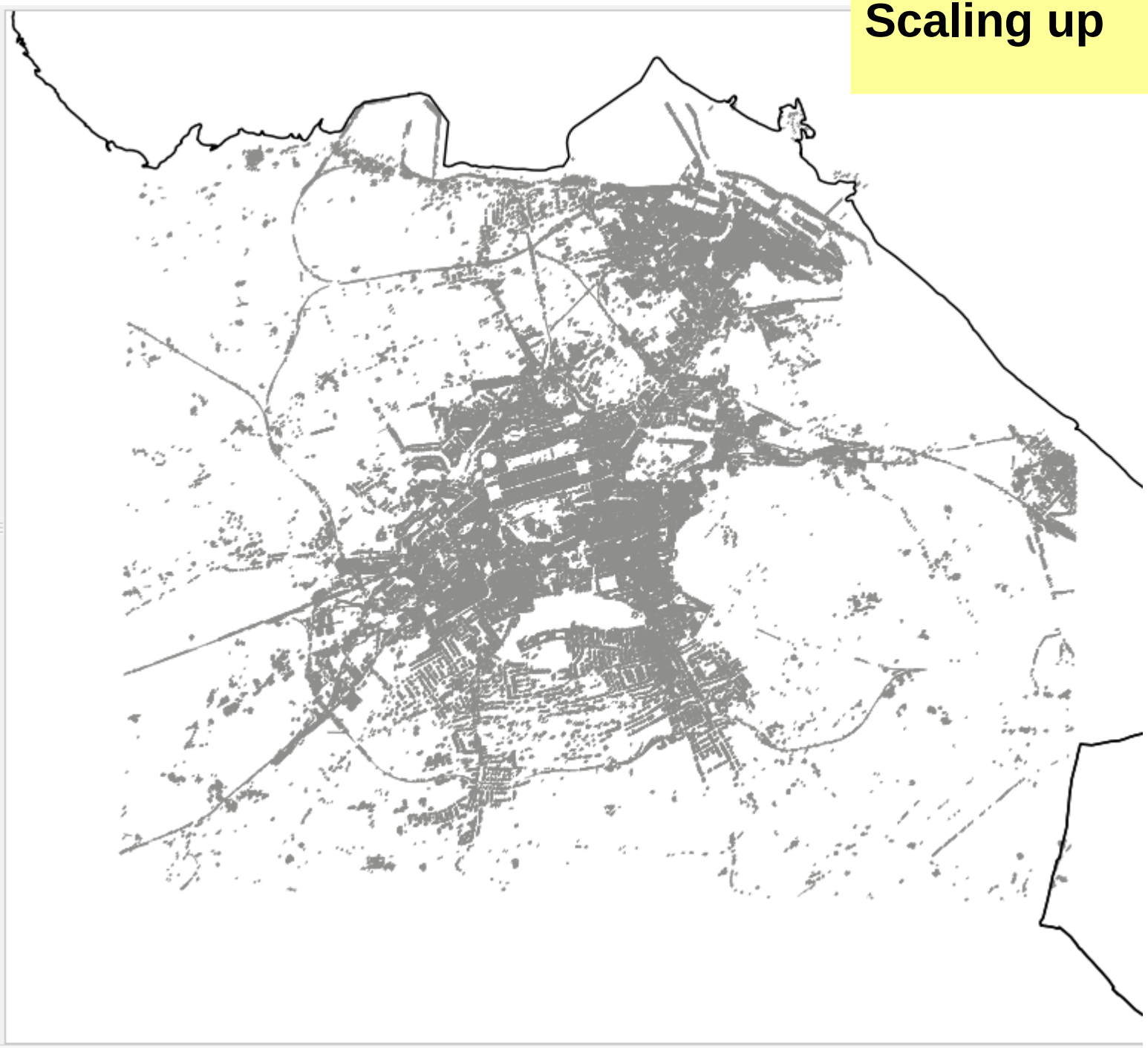




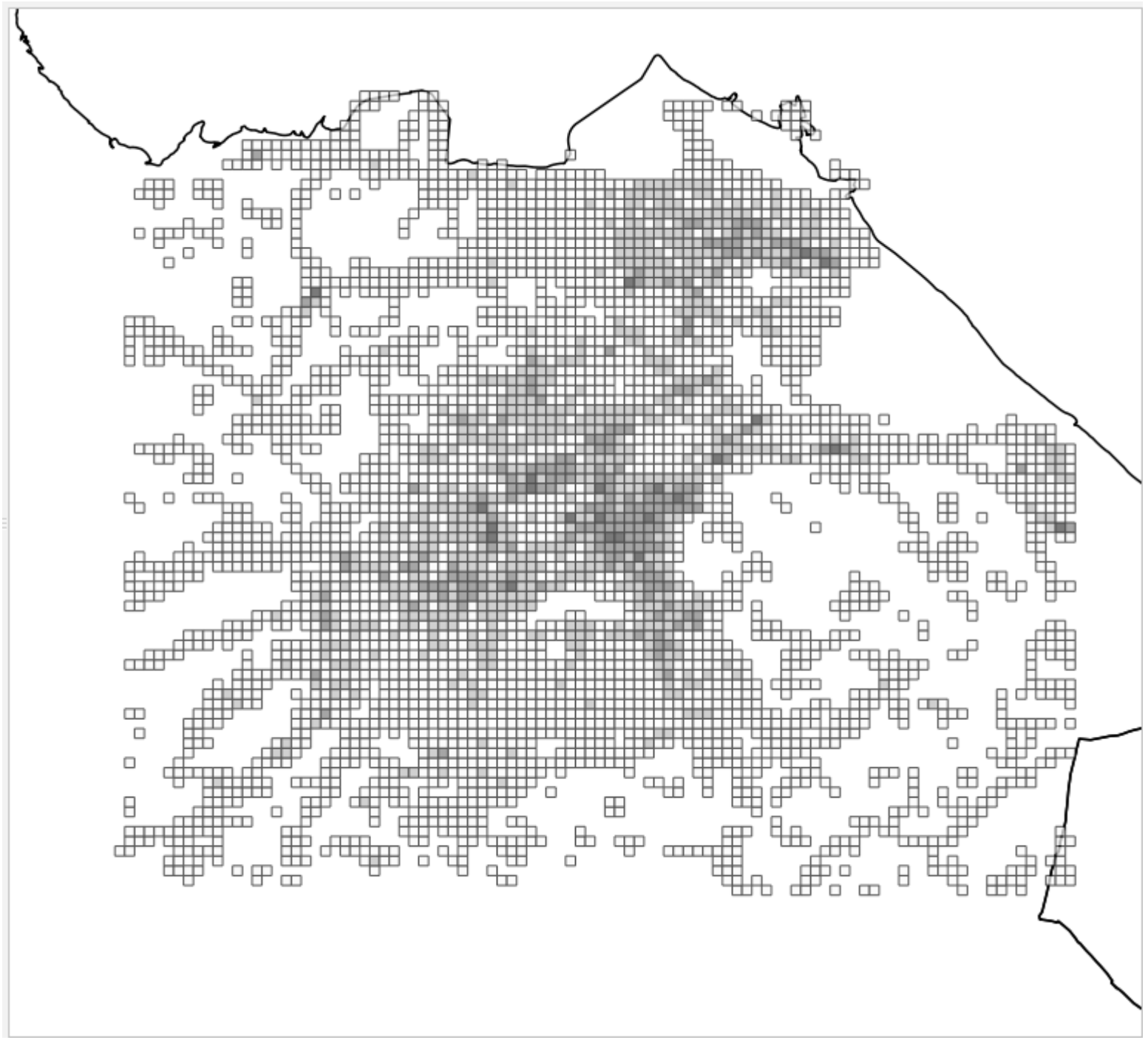


End product would be a grid describing % building (built-up) across each 100m x 100m standard grid square in ca1900. Data could be aggregated upwards e.g. to produce a 1km x 1km grid. Using the same sampling grid could compute the same measure for modern data (I`ve used OS MasterMap but other OS OpenData could be used). Could then calculate + / - change between ca1900 and today / other time periods for which historic maps available.

Scaling up



Process repeated for whole of Edinburgh using all 19 NLS map sheets – urban form of Edinburgh ca1900.



Same 100m x 100m grid across Edinbrugh as a whole in ca1900

Edinburgh % Building ca1900 (Historic map)
- 0.0 - 20.0
- 20.0 - 40.0
- 40.0 - 60.0
- 60.0 - 80.0
- 80.0 - 100.0