**Determining Candidate Cell Sites**

Table of Contents

[**Steps to Creating to Create Candidate Cell Sites** 2](#_Toc194486570)

[**Finding the Highest Elevation Points** 3](#_Toc194486571)

# **Steps to Creating to Create Candidate Cell Sites**

1. Filter expression: “Id” != <change into #> = 0
2. Clip raster by buffer
3. Raster pixels to points
4. Determine **10 of the highest elevation points**
5. Clip road by buffer
6. Determine **5 of the 10 highest eleveation points are closest to roads**
7. Clip buildings by buffer
8. **5 most centrality nodes**
9. Determine the one candidate cells that is the most central point with relation to points as buildings

# **Finding the Highest Elevation Points**

|  |
| --- |
| **PyQGIS / Python Code** |
| **from qgis.core import (**  **QgsProject, QgsSpatialIndex, QgsFeatureRequest, QgsWkbTypes**  **)**  **# -- Parameters: adjust layer names and thresholds as needed – #**  **point\_layer\_name = ‘ClippedRasterPoints’**  **road\_layer\_name = ‘ClippedRoadNetwork’**  **building\_layer\_name = “ClippedBuildings”**  **#Distance thresholds (in meters) for “closeness”**  **road\_threshold = 50 #maximum allowed distance for a road**  **building\_threshold = 50 # maximum allowed distance for a building**  **# Name of the elevation attribute in the point layer**  **elevation\_attr = 'VALUE'**  **# --- Get the layers from the current QGIS project ---**  **project = QgsProject.instance()**  **point\_layers = project.mapLayersByName(point\_layer\_name)**  **road\_layers = project.mapLayersByName(road\_layer\_name)**  **building\_layers = project.mapLayersByName(building\_layer\_name)**  **if not point\_layers or not road\_layers or not building\_layers:**  **raise Exception("One or more layers not found. Please check your layer names.")**  **point\_layer = point\_layers[0]**  **road\_layer = road\_layers[0]**  **building\_layer = building\_layers[0]**  **# --- Build spatial indexes for roads and buildings for fast nearest-neighbor lookup ---**  **road\_index = QgsSpatialIndex(road\_layer.getFeatures())**  **building\_index = QgsSpatialIndex(building\_layer.getFeatures())**  **# --- Loop through each point feature and compute distances ---**  **qualifying\_points = [] # to hold tuples: (feature\_id, elevation, road\_dist, bldg\_dist, feature)**  **for feature in point\_layer.getFeatures():**  **geom = feature.geometry()**  **# Check: ensure the geometry is a point.**  **# We use QgsWkbTypes to get the type of geometry.**  **if not geom or QgsWkbTypes.geometryType(geom.wkbType()) != QgsWkbTypes.PointGeometry:**  **continue**  **#Get the point coordinates**  **pt = geom.asPoint()**  **#Get nearest road feature**  **road\_ids = road\_index.nearestNeighbor(pt, 1)**  **if road\_ids:**  **# Use a feature request to get the nearest road feature by ID**  **road\_req = QgsFeatureRequest(road\_ids[0])**  **road\_feature = next(road\_layers.getFeatures(road\_req))**  **road\_distance = geom.distance(road\_feature.geometry())**  **else:**  **road\_distance = float(‘inf’)**  **# Get nearest building feature**  **building\_ids = building\_index.nearestNeighbor(pt, 1)**  **if building\_ids:**  **building\_req = QgsFeatureRequest(building\_ids[0])**  **building\_feature = next(building\_layers.getFeatures(building\_req))**  **building\_distance = geom.distance(building\_feature.geometry())**  **else:**  **building\_distance = float(‘inf’)**  **# Only consider points that are close enough to both roads and buildings.**  **if road\_dist <= road\_threshold and bldg\_dist <= building\_threshold:**  **# Get the elevation value from the attribute table.**  **try:**  **elev = float(feat[elevation\_attr])**  **except (ValueError, TypeError):**  **continue**    **qualifying\_points.append( (feat.id(), elev, road\_dist, bldg\_dist, feat) )**  **# --- Sort the qualifying points ---**  **# Primary sort: highest elevation (descending)**  **# Secondary sort: smallest total distance to road and building (ascending)**  **qualifying\_points.sort(key=lambda x: (-x[1], x[2] + x[3]))**  **# --- Select the top three points ---**  **selected\_ids = [pt[0] for pt in qualifying\_points[:3]]**  **if selected\_ids:**  **point\_layer.selectByIds(selected\_ids)**  **print("Selected point feature IDs:", selected\_ids)**  **else:**  **print("No points found that meet the criteria (within thresholds and with a valid elevation).")** |

# **Candidate Cell Site Algorithm**

## **Determining the candidate cell sites closest to buildings**

### **Algorithm**

1. Create a line from the candidate cell to a building point.
2. Measure that line
3. Repeat for all other building points
4. Sum up all the measurements
5. candidateCellSites\_wMeasurement 🡨 {}
6. FOR candidate\_cellSite in candidate\_cellSites:
7. candidateCellSites\_wMeasurement[candidate\_cellSite] = 0
8. FOR building\_point in building\_points:
9. Line 🡨 create\_line(candidate\_cellSite, building\_point)
10. Temp\_measurement = measure(line)
11. candidateCellSites\_wMeasurement[candidate\_cellSite] += Temp\_measurement
12. Temp = INFINITY
13. closest\_cells = []
14. FOR key, value in candidateCellSites\_wMeasurement.items():
15. if value < Temp:
16. Temp 🡨 value
17. Closest\_cells.clear()
18. Closest\_cells.append(key)
19. Elif value = Temp:
20. Closest\_cells.append(key)
21. Return closest\_cells

## **Determining the candidate cell sites with the highest elevation**

### **Algorithm**

1. Clipped the raster file relative to the cell
2. Convert raster pixels into points
3. Determine point with the highest value

## **Determine the candidate cell sites closest to roads**

### **Algorithm**

1. A

## **Determine the candidate cells sites closest to the circumferential road**

### **Algorithm**