Ranplan Algorithm Engineer Test
Radio Propagation Model

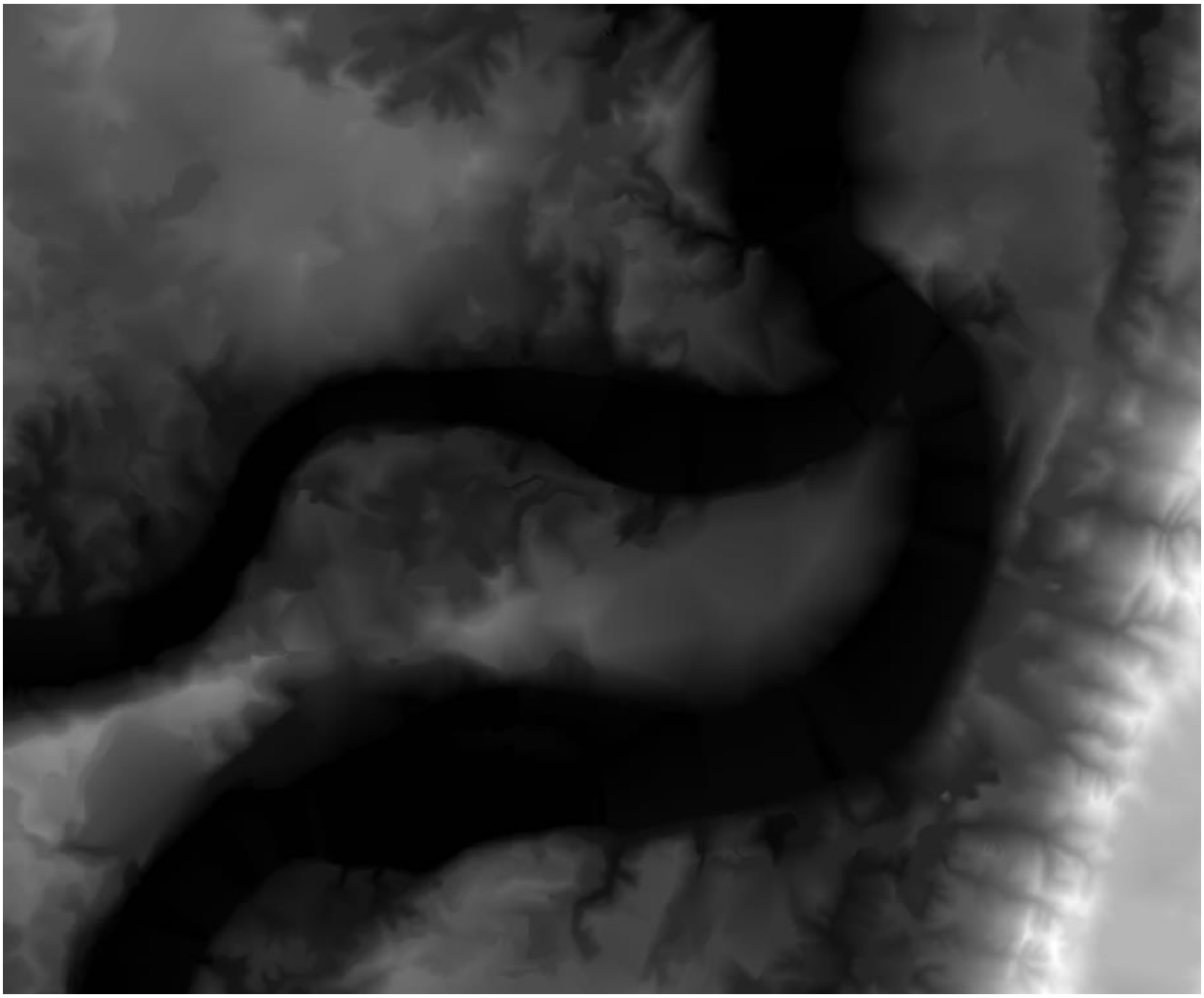
Data Description

You will receive a sample of terrain data, which is in a raster format (e.g. image) with a specific width and height. It also comes with a resolution to define its exact distance between two pixels.

The spec table is defined below:

Field Name	Value	Units
Name	terrain.bil	
Data Type	Signed Integer 16	
Resolution	5	metre
Rows	1396	
Columns	1689	

Sample data reference image is illustrated below.



Radio signal will propagate from a transmitter (i.e. base station) to a receiver (i.e. mobile phone), the signal degradation between the wireless connection is called “path loss”.

In this test, we are going to estimate the path loss based on two common models (i.e. radio propagation model).

The output data (i.e. path loss) will be saved in a similar format as the input with same resolution and rows/columns profile. When multiple heights H are considered at the a given pixel (X, Y), it is required to save the data as multiple tables in the same binary file. In this test, we will use max_height to define the maximum height, the calculation height range will be equal to {resolution: resolution: max_height}. Example: max_height = 50, resolution = 5, H = {5, 10, 15, 20, 25, 30, 35, 40, 45, 50}. Please notice, H is determined as the receiver height above the terrain.

Interface Description

A Command Line Interface (CLI) tool will be required to run the test. The syntax will look similar as:

Syntax:

Model_Name terrain.bil resolution rows columns max_height transmitter_x transmitter_y, transmitter_h transmitter_freq output_file_name

Where:

- *Model_Name: name of your propagation model (i.e. your executable name)*
- *terrain.bil: name of the given terrain input data*
- *resolution: resolution of the given terrain input data*
- *rows/columns: number of rows and columns of the given terrain data*
- *max_height: the maximum of the receiver height*
- *transmitter_x/transmitter_y: the x, y coordinates in metre relative to 0,0 pixel*
- *transmitter_h: height of the transmitter in metre*
- *transmitter_freq: Signal frequency of the transmitter in MHz*
- *output_file_name: File name of the output data*

You will then need to implement the following two common radio propagation models to generate the path loss output data.

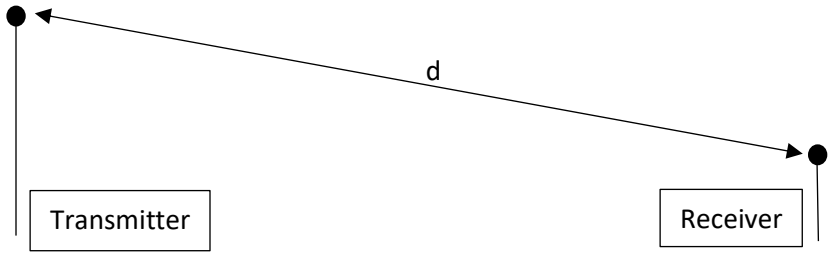
Free Space Path Loss (FSPL) Model

The equation below shows the path loss for a free space propagation application.

$$FSPL(dB) = 20\log(d) + 20\log(f) + 32.45$$

Where:

- d: distance of the receiver from the transmitter (**km**)
- f: signal frequency (**MHz**)
- FSPL: free space path loss (**dB**)



Please use the FSPL model to calculate the path loss and save to an output file

You CLI may look like below for the sample data:

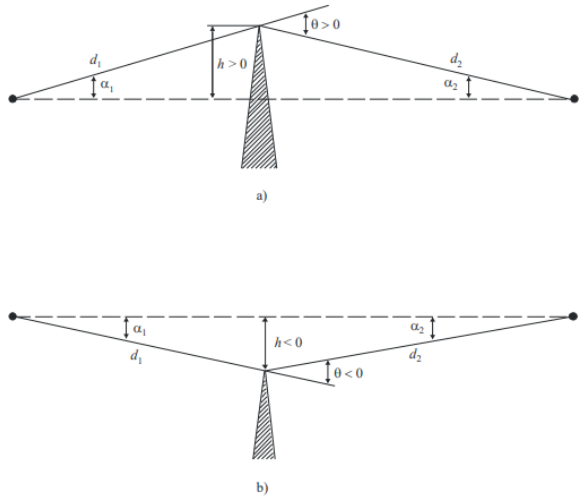
Syntax:

FSPL terrain.bil 5 1396 1689 max_height transmitter_x transmitter_y transmitter_h transmitter_freq fspl.pl

Knife Edge Diffraction (KED) Model

When signal is blocked by the terrain, extra loss will be considered in addition to the free space loss. One of the common methods is so called knife edge diffraction (KED) model.

For a single edge model (when there is only one blockage between the transmitter and the receiver), the formula is given by:



$$\nu = h\sqrt{\frac{2}{\lambda}\left(\frac{1}{d_1} + \frac{1}{d_2}\right)}$$

$$J(\nu) = 6.9 + 20\log\left(\sqrt{(\nu - 0.1)^2 + 1} + \nu - 0.1\right) \quad \text{for } \nu \geq -0.7$$

$$J(\nu) = 0 \quad \text{for } \nu < -0.7$$

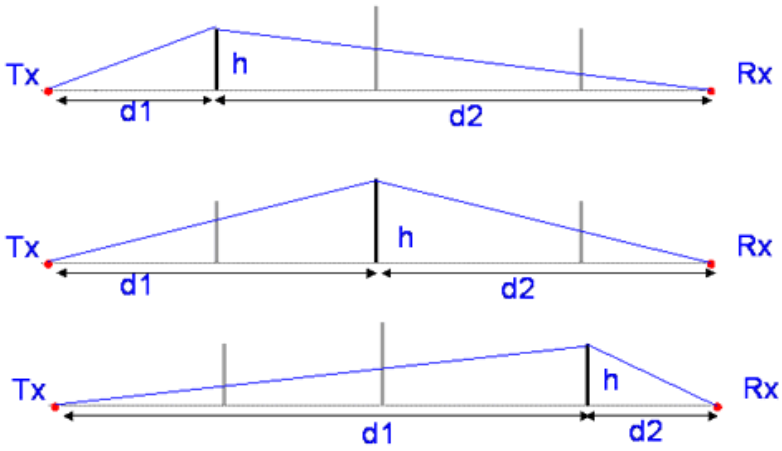
Where:

- h: height between the edge and transmitter-receiver edge (**m**)
- d1, d2: distance between the edge and transmitter, receiver correspondingly (**m**)
- λ: signal wavelength (**m**) (i.e. v/f)
- J(v): diffraction loss in additional to the free space loss (**dB**)

Multiple Edges

When there are multiple edges between the transmitter and receiver, multi-edge diffraction models will be used. One of the common method is called Deygout Method (Propagation Tutorial - Diffraction

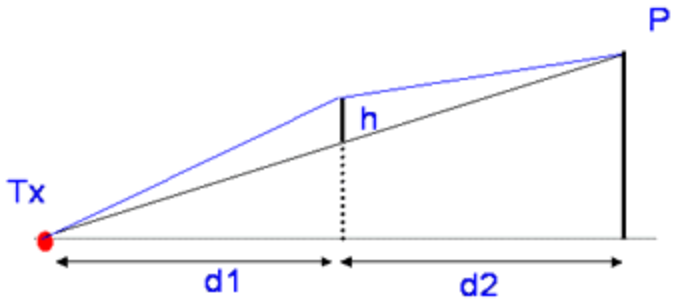
(mike-willis.com)). It splits the path into segments. Firstly we need to find edge with largest value of parameter v, ignoring all other edges. This is called the “Principle Edge” and its v parameter is saved.



$$v = h\sqrt{\frac{2(d_1 + d_2)}{\lambda d_1 d_2}}$$

E.g Say it is this edge

Now working from the principle edge P, we treat as if there is a new path between the transmitter and the principle edge and create a new reference plane and calculate v for the intermediate edge, if there is one, based on the height above the reference plane.



This edge will have a lower value of v and becomes the principle edge for the path from transmitter to P. The process is **recursive** for multiple intermediate edges and can be repeated until all edges are considered. The same process is used along the path from P to the receiver.

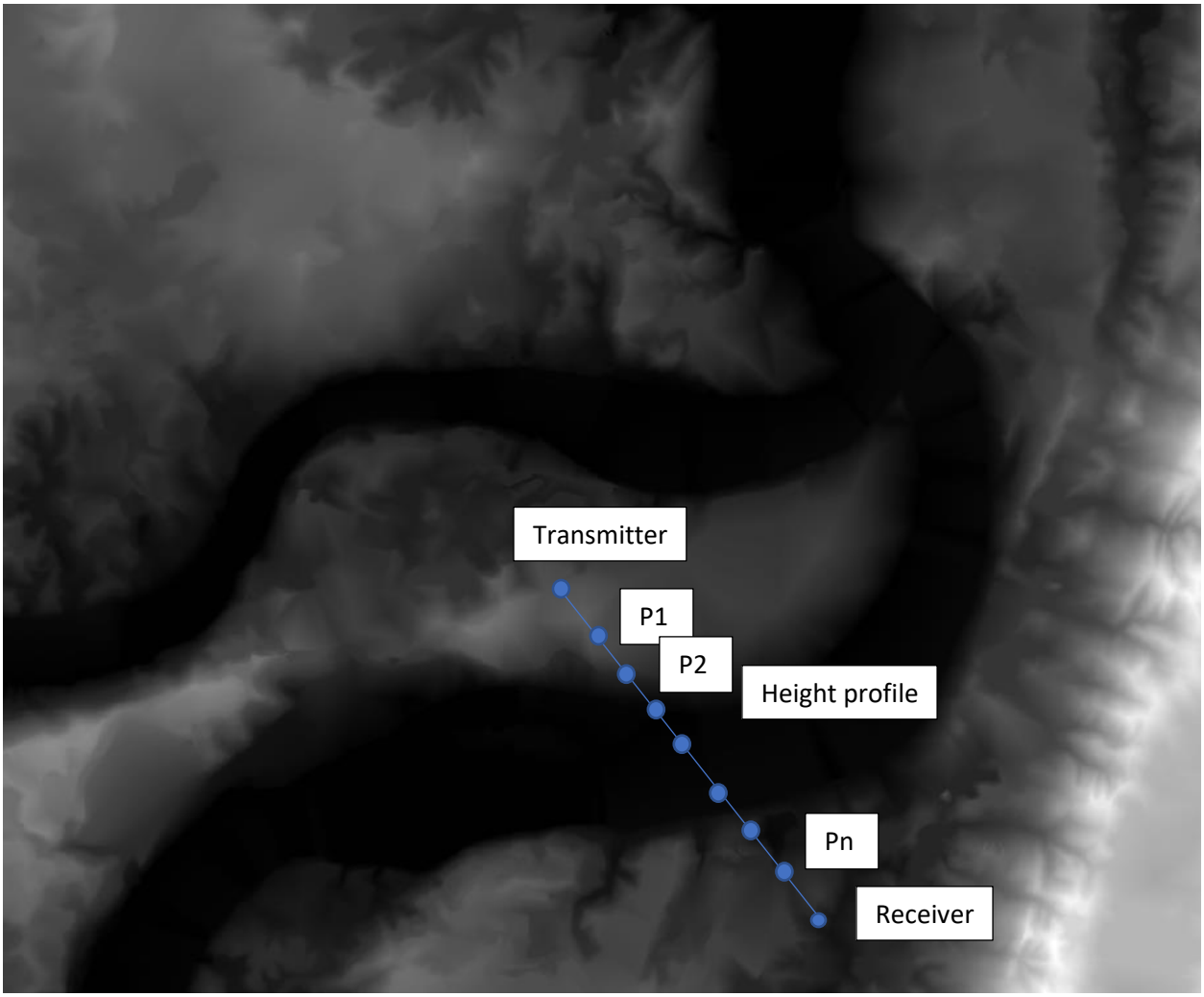
At the end of the procedure, we will have a set of J(v) losses for each edge - the method simply adds these up. So for 3 edges:

L = J(vp) + J(vtp) + J(vpr)

KED = FSPL + L

The common implementation logic would be divided into two stages,

- Stage A. for a given Transmitter location and Receiver location, a terrain height profile can be extracted between the two nodes: h_profile = {P1, P2, …, Pn}
- Stage B. for given specs of the transmitter {X, Y, H, Freq} and receiver {x, y, h}, together with the height profile h_profile, calculate the free space loss plus the corresponding diffraction loss (if any).



Please note: KED calculation efficiency will be the most important testing criteria for your algorithm design capability.

You CLI may look like below for the sample data:

Syntax:

KED terrain.bil 5 1396 1689 max_height transmitter_x transmitter_y transmitter_h transmitter_freq ked.pl

Random Terrain Data Generator

This task is to write a CLI to generate random terrain automatically. Make sure your data can work with the two radio propagation models.

You CLI may look like below for the sample data:

Syntax:

RTDG terrain.bil resolution rows columns min_terrain_height max_terrain_height additional_para

Where:

- *RTDG: name of your random terrain generator (i.e. your executable name)*
- *terrain.bil: name of the given terrain input data*
- *resolution: resolution of the given terrain input data*
- *rows/columns: number of rows and columns of the given terrain input data*
- *min_terrain_height/max_terrain_height: the range of the terrain height*
- *additional_para: you can add more parameters to make the terrain data looks more realistic*

Summary of the Test

Programming Languages

You can finish you tests in Java, Python, C/C++ or .net.

Test Deliverables

Your deliverables will include:

D1. Free Space Propagation Model (FSPL)

D2. Knife Edge Diffraction Model (KED)

D3. Random Terrain Data Generator (RTDG)

Test Ranking

Your score will be based on both the accuracy of the results and efficiency of your algorithm.