Detecting gullies in LIDAR bare earth elevation data

Jim Pivarski

University of Chicago – Data Science Institute

October 31, 2025

What I found in the literature

Several papers all described the same technique:

- draw images of small-scale variations in elevation by subtracting elevation at each point with the average of its neighbors.

What I found in the literature

Several papers all described the same technique:

- draw images of small-scale variations in elevation by subtracting elevation at each point with the average of its neighbors.

Here's one:

High resolution quantification of gully erosion in upland peatlands at the landscape scale

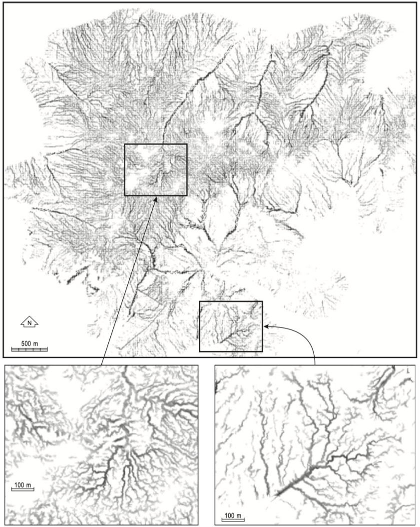
Martin Evans[1]* and John Lindsay[2]
[1] Upland Environments Research Unit, School of Environment and Development, University of Manchester, Manchester, UK
[2] Department of Geography, University of Guelph, Guelph, Belgium

Received 24 March 2009; Revised 19 August 2009; Accepted 3 September 2009

*Correspondence to: Martin Evans, Upland Environments Research Unit, School of Environment and Development, University of Manchester, Manchester M13 9PL, UK. E-mail: martin.evans@manchester.ac.uk

What I found in the literature

Gullies occupy low local topographic positions. Several terrain attributes can be calculated using digital elevation data to assess local lowness. Difference from mean elevation (DFME) is the difference between the elevation at the centre of a moving filter window and the mean elevation in the window (Gallant and Wilson, 2000). DFME is commonly used for elevation residual analysis to express the pattern of local topographic position. This terrain attribute can be negative (i.e. lower than the local average elevation) or positive (i.e. higher than the average elevation) and is measured in units of length. It can be calculated in a two-step process by filtering the DEM using a mean filter and differencing the result from the original DEM. DFME is equivalent to a high-pass filter in the field of image processing and can be calculated using most standard GISs and image processing software. Similar terrain

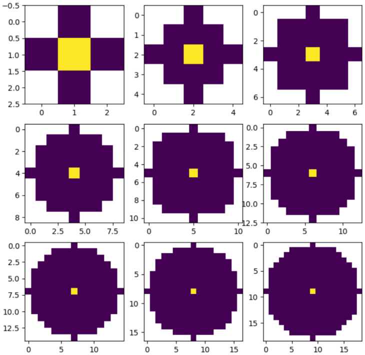


This procedure is a convolution with an appropriately defined kernel

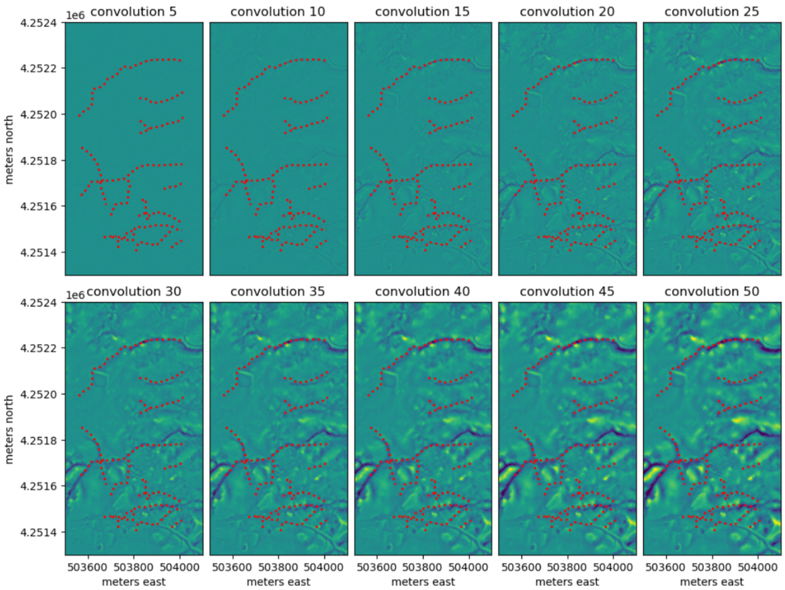
"Subtract each pixel from the average of a disk centered on that pixel."

is equivalent to

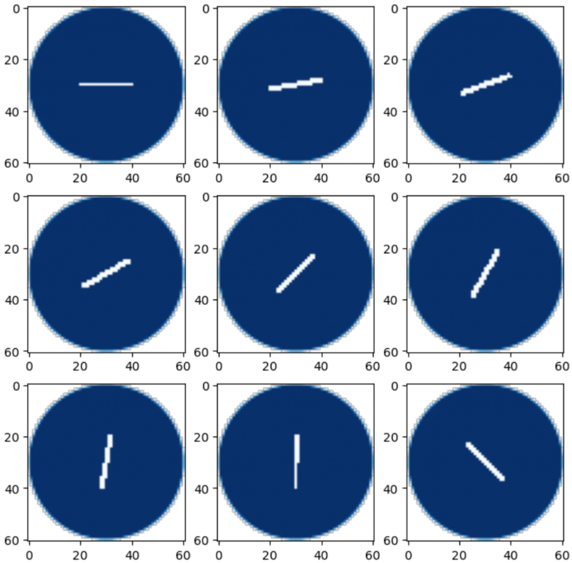
"Convolve with a kernel that is positive in the center pixel, uniformly negative in a disk around the pixel, zero elsewhere, and sums to zero."

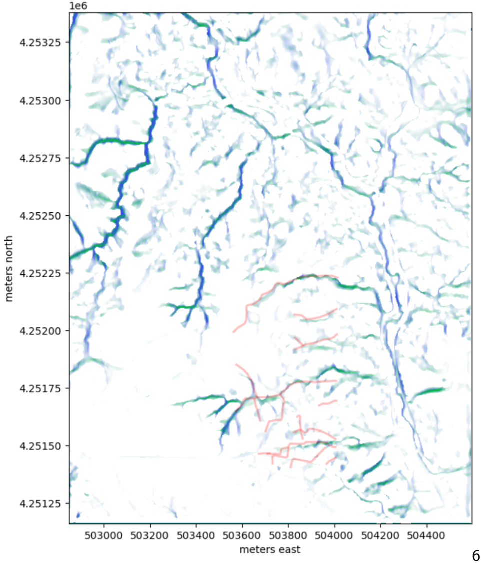


It works! Especially if the disk is very large



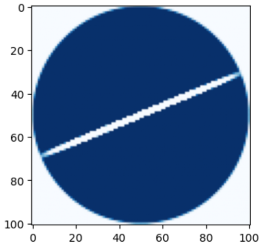
But if a point-in-disk is good, wouldn't a line-in-disk be better?



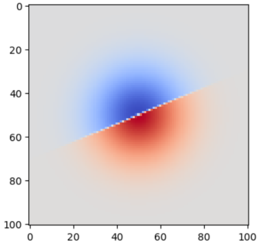


I studied the response of gully and non-gully pixels to various kernels

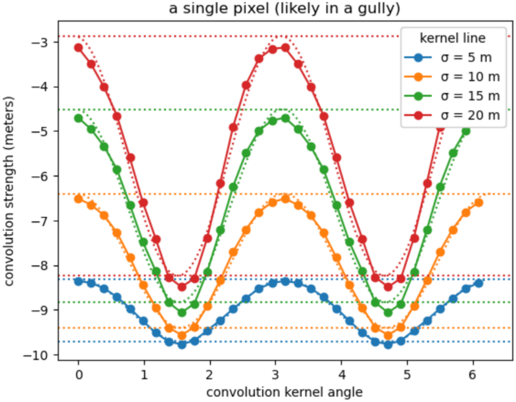
A refined **line-in-disk kernel** in which the line is a Gaussian peak with width σ.



This **cliff-edge kernel** responds best to a sharp drop in elevation, aligned with its angle.



Gully pixels respond best when the line-in-disk kernel's angle is aligned with the gully trough. Also, the parallel-versus-perpendicular difference is biggest for the longest lines.



Optimizing a gully-finder model

- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.

Optimizing a gully-finder model

- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.
- Minimum difference between high and low (varying angle) is a good indicator of *not* being in a gully.

Optimizing a gully-finder model

- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.
- Minimum difference between high and low (varying angle) is a good indicator of *not* being in a gully.
- Empirically, real gullies are a little more pointed than a sinusoidal fit.

Optimizing a gully-finder model

- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.
- Minimum difference between high and low (varying angle) is a good indicator of *not* being in a gully.
- Empirically, real gullies are a little more pointed than a sinusoidal fit.
- The line-in-disk kernel matches one-sided cliff-edges too well, but the specialized cliff-edge kernel matches cliffs and not gullies, so it can be a veto.

Optimizing a gully-finder model

- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.
- Minimum difference between high and low (varying angle) is a good indicator of *not* being in a gully.
- Empirically, real gullies are a little more pointed than a sinusoidal fit.
- The line-in-disk kernel matches one-sided cliff-edges too well, but the specialized cliff-edge kernel matches cliffs and not gullies, so it can be a veto.

$\longrightarrow$ all of these can be features in an optimized model.

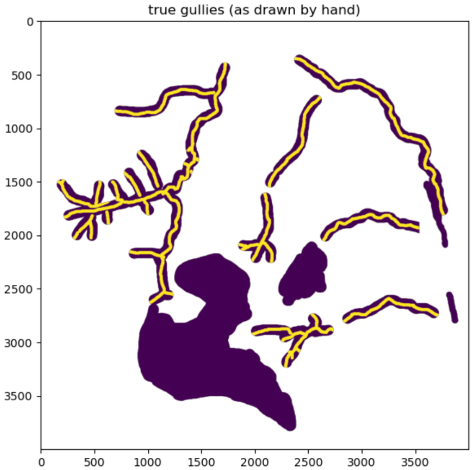
Optimizing a gully-finder model

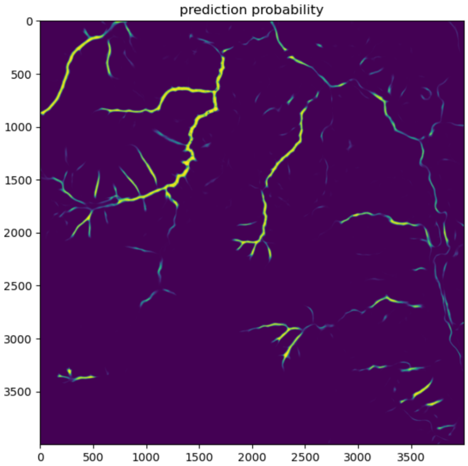
- Maximum convolution response (maximized over angle) is a good indicator of being in a gully.
- Minimum difference between high and low (varying angle) is a good indicator of *not* being in a gully.
- Empirically, real gullies are a little more pointed than a sinusoidal fit.
- The line-in-disk kernel matches one-sided cliff-edges too well, but the specialized cliff-edge kernel matches cliffs and not gullies, so it can be a veto.

$\longrightarrow$ all of these can be features in an optimized model.

Since there's only a handful of features, it will be a linear fit.

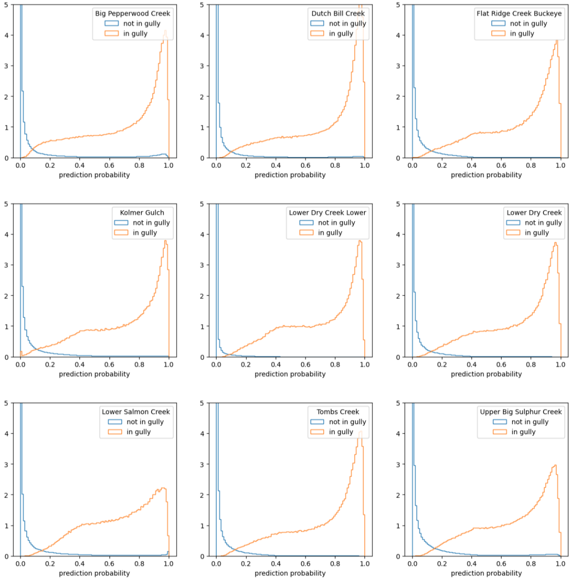
Supervised learning for the linear model





I randomly selected 9 watersheds and drew yellow on what I knew were gullies (with help from Google Street View), blue on non-gullies, and I left the rest. The fitter's job is to optimize the linear combination so that it makes the same decisions I did.

Fit results



$\longleftarrow$ Prediction on training data (loss)

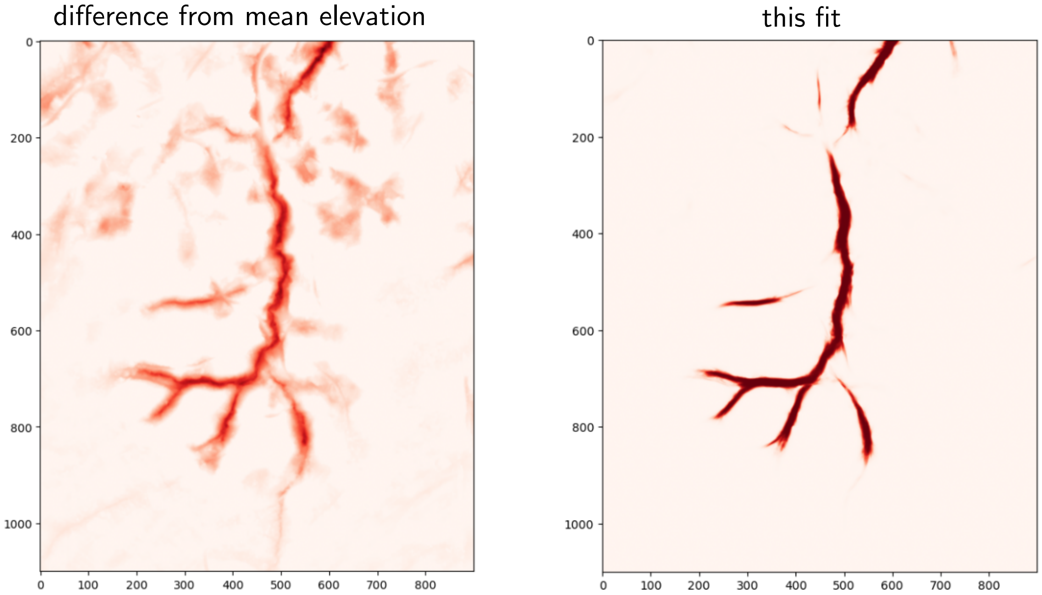
Best-fit parameters:

-9.80	`(intercept)`
-3.16	`min15`: best line-in-disk
-0.72	`low15 − min15`: non-sinusoidal
0.20	`min5 − min15`: dependence on σ
1.94	`high15 − low15`: on angle
-0.27	`high5 − low5`: on angle
0.63	`low15 * (high15 − low15)`
-0.29	`abs(mindisk)`: cliff-edge veto

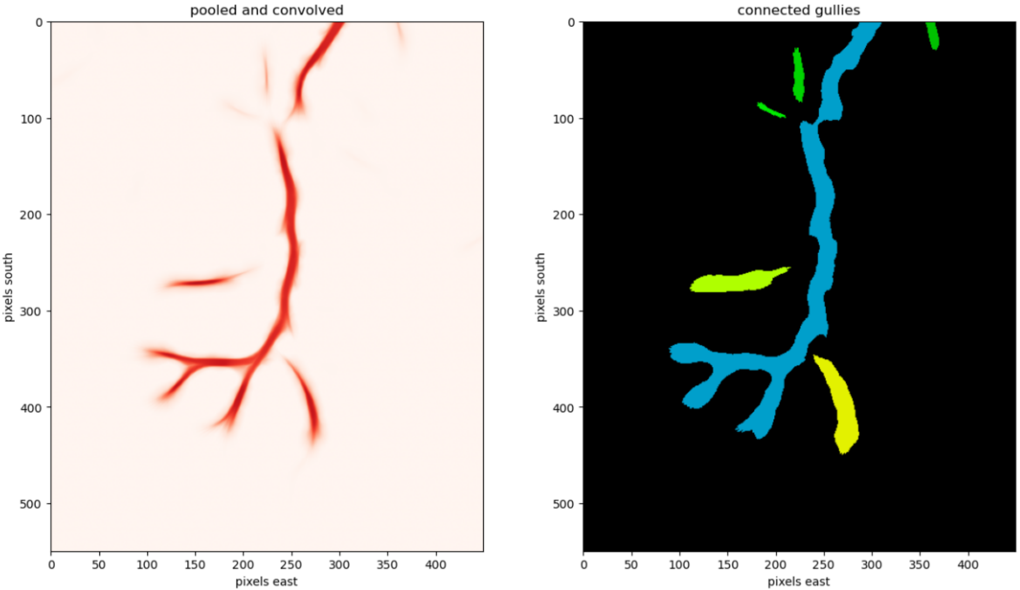
Parameter correlations:

	(intercept)	min15	low15 - min15	high15 - low15	min5 - min15	high5 - low5	low15 * (high15 - low15)	abs(mindisk)
(intercept)	1.000000	0.854740	0.086704	-0.472281	-0.000096	-0.011087	-0.825057	0.029504
min15	0.854740	1.000000	0.261732	-0.094373	-0.297184	-0.241753	-0.738537	0.285971
low15 - min15	0.086704	0.261732	1.000000	0.219852	-0.262007	-0.499590	-0.057264	0.130928
high15 - low15	-0.472281	-0.094373	0.219852	1.000000	-0.625818	-0.619049	0.643569	0.038105
min5 - min15	-0.000096	-0.297184	-0.262007	-0.625818	1.000000	0.473387	-0.190269	-0.150616
high5 - low5	-0.011087	-0.241753	-0.499590	-0.619049	0.473387	1.000000	-0.076208	-0.095644
low15 * (high15 - low15)	-0.825057	-0.738537	-0.057264	0.643569	-0.190269	-0.076208	1.000000	-0.127800
abs(mindisk)	0.029504	0.285971	0.130928	0.038105	-0.150616	-0.095644	-0.127800	1.000000

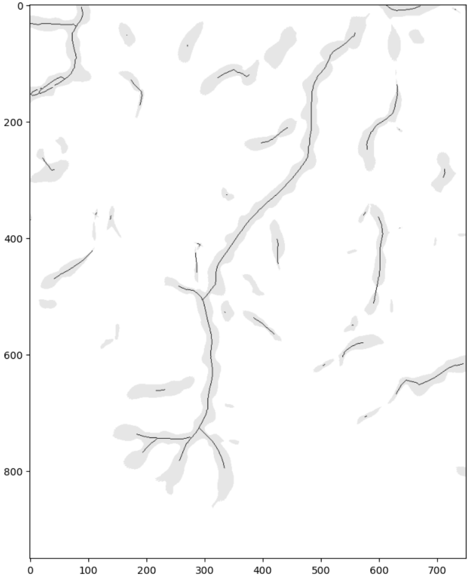
Does it matter? How does it compare with the state-of-the-art?



Beyond the fit: turning the raster into a graph



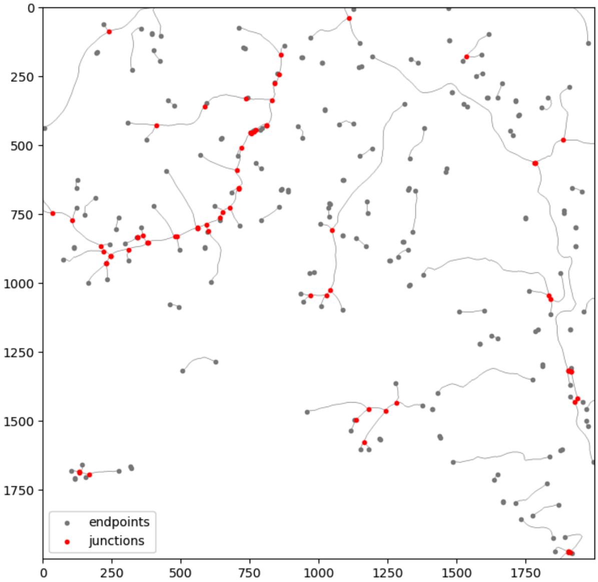
Skeletonizing the graph



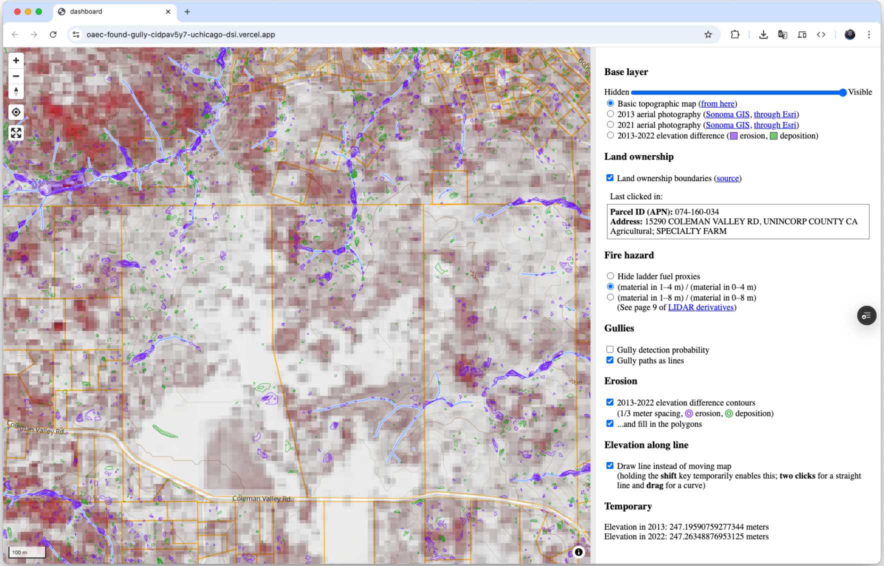
- Take all pixels with $P > 0.01$ to be nodes of a graph, potentially connected to 8 neighbors (NW, N, NE, W, E, SW, S, SE).
- In order of increasing probability P, remove a node if removing it does not disconnect its connected component (greedily).

By construction, each connected component has no cycles, which also implies that it is a single node wide.

Gullies are now like a road network

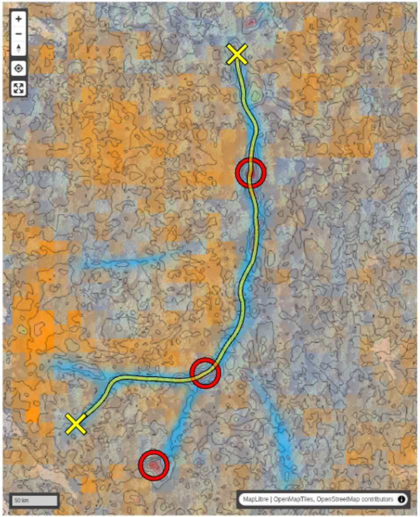


. . . that I'm putting on a crowded map



Goal: make elevation differences queryable along gullies

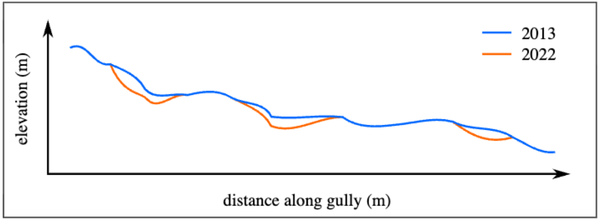
Dutch Bill watershed



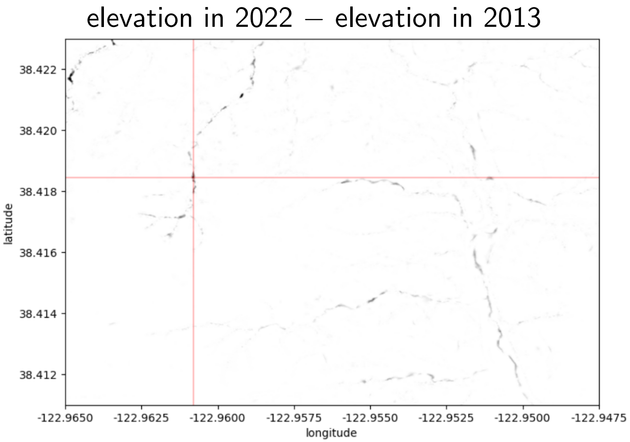
Alerts:

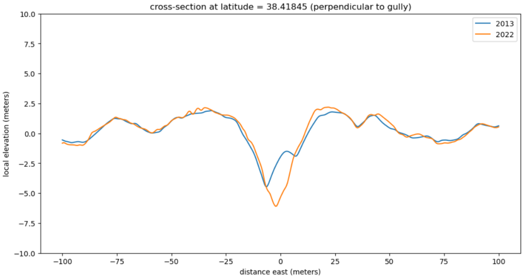
Longitude	Latitude	Erosion depth (m)	Proximity to ladder fuels

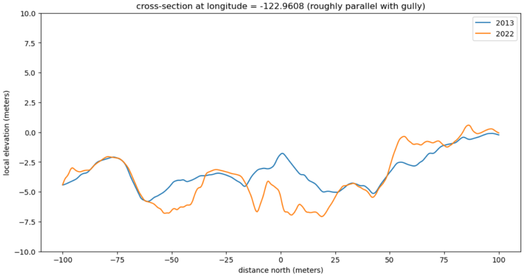
Elevation along gully:



Because 2013–2022 LIDAR differences along gullies show erosion!







Back to the map app (in development)