In the above curve has pixels and every pixel has a value associated with it which can be represented using some no. of bits like 16 bits. It can be a terrain where each pixel has some height.

Aim – To transmit a level curve

To find a level Curve \( C(z) = \{ p : h(p) = z \} \)

Where \( p = (i, j) \) is a continuous set of points
\[ z = \text{height} \]

Here we are going to assume that client has very less memory and can store only one level curve.

For each cell we find whether curve intersects it or not.

We interpolate height in the cell. For this we can use bilinear interpolation.
Algorithms to find C(z) →

**Algo 1** (Not very efficient) : For each cell C compute its contribution to C(z) and send it.

Different configurations in which a level curve can intersect a cell in:

+ : the point is at height more than the level curve
- : the point is at height lower than the level curve
We use linear interpolation to find entry and exit points.

This algorithm sends the whole curve. But we don’t want to sent the whole curve because of the limited amount of resources available to the client

So the foll algorithm is more efficient than this one.

**Algorithm 2:**

Here we don’t send all the points. Here after finding the initial point on curve depending on the height of the corners of the cell (+ / --) we only transmit L (for going left), R (for going right) or S (for going straight).

We always move by keeping the point at more height on the left.