EZCOD Geolocation Encoding

Perry Kundert

2014-06-25 00:00:00

Quickly and accurately encode and decode geolocations in your own C++, Javascript or Python program.

Source at `https://github.com/pjkundert/ezpwd-reed-solomon`. Get more information at `https://dominionrnd.com/products/13-reed-solomon` (PDF)

Contents

1 EZCOD 3:10 – 3m accuracy in 10 symbols

1.1 Why Another Geo-Location Encoding?

Google's proposed Open Location Code enumerates a comprehensive evaluation of location encoding systems, including a detailed list of desired attributes:

- Codes should be short enough that they can be memorised
- A code should be sufficient on it's own, not requiring additional information such as country name
- To prevent confusion, a place should have only one code
- Codes should not include easily confused characters (e.g., 0 and O, 8 and B etc)
- Codes should not include profanity, or preferably, any words in any language
- It should be possible to look at two codes and tell if they are close together and even the direction. This will help people find their way and make the codes more usable;
- Codes should represent an area, not a point, where the size of the area is variable

- Shortening a code should represent a larger area that contains the original location. A side affect of this is that nearby codes will have a common prefix;
- The code for a place should be deterministically generated, not requiring any setup or application
- Codes should be able to be generated and decoded offline. Mobile data networks may not be ubiquitous or cheap
- Codes should not depend on any one provider, so that there is no risk that they will stop working if one company goes out of business
- The algorithm should be published publicly and be free to use.
- A code should be sufficient on it's own, not requiring additional information such as country name

These are excellent goals, and EZCOD implements all except one (impossible to implement) feature. However, in this list, the most important feature is missing. This is where EZCOD excels vs. all existing encoding schemes:

- Detects errors and is self-correcting.

1.2 Error Resilience

Geolocations are rarely so unimportant that an assurance of accuracy is not desirable. Most existing geolocation encodings have no self-verification features at all, and those that do can only detect errors (a single check character, with low reliability), and cannot correct any errors.

EZCOD implements Reed-Solomon encoding, and even the most basic form can both detect and correct a loss of data.

1.3 Importance of Error Detection

To specify the location of something on the surface of the earth, a Latitude, Longitude pair is typically used. To get within +/-3m, a Latitude, Longitude pair with at least 5 digits of precision after the decimal point is required. This is the default precision of EZCOD, with 9 symbols of location and 1 symbol of Reed-Solomon parity (separated by an optional . or ! symbol).

So, to specify where my daughter Amarissa was born, I can write down the coordinate:

```
53.655832,-113.625433
```

The equivalent EZCOD is:

```
R3U 1JU QUY.0
```

If an error is introduced (a symbol is lost, indicated by the _) in the numeric encoding, the difference in position may be anywhere from centimeters to many kilometers:

```
53.655832,-113.62543_ == a few centimeters error
53.655832,-1_3.625433 == many kilometers error
```

If a symbol is similar symbol is lost in an EZCOD, the standard encoding with a single Reed-Solomon parity symbol can fully recover the position, yielding 100% accuracy:

```
R_U 1JU QUY.0 == no error
R3U 1JU QUY._ == no error
```

It can also reliably detect a single erroneous symbol with 100% certainty, and multiple erroneous symbols with reliability at least as good as any other single check-character system:

```
R_U 1JU QUY._ == too many erasures
RXU 1JU QUY.X == R-S decode overwhelmed
```

1.4 Variable Error Detection/Correction Capacity

There are often scenarios where higher levels of certainty are desirable. By specifying additional Reed-Solomon parity symbols, much higher reliability can be achieved. Using 3 parity symbols yields almost 5-nines certainty, P(0.99997).