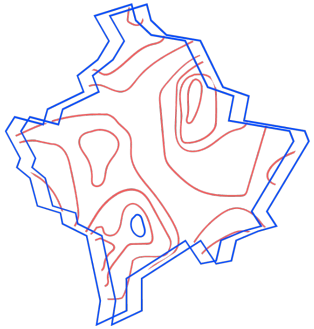
source

Site Calibration

with PROJ and WKT2

Site Calibration

with PROJ and WKT2



FOSS4G

Prizren, 2023

Javier Jimenez Shaw

PROJ contributor.

Civil Engineer and Software Developer.

Technical Coordinator of SRS team at Pix4D.

https://github.com/jjimenezshaw/

Site calibration with PROJ and WKT2*

The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-4/W7-2023
FOSS4G (Free and Open Source Software for Geospatial) 2023 – Academic Track, 26 June–2 July 2023, Prizren, Kosovo

Purpose of the paper:

There are some proprietary solutions (“black box”, not well documented) to the site calibration.

- We offer a solution using an open format and open source libraries.

* No AI was harmed in the making of this paper.

ABSTRACT:

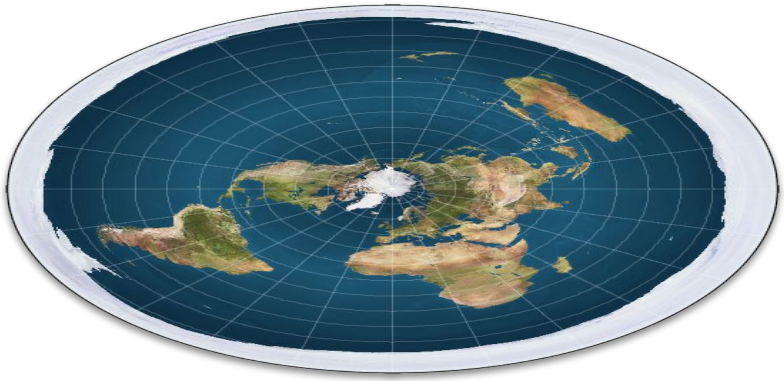
In surveying projects, a site calibration is the solution to the problem of finding a transformation between the coordinate space of a site local coordinate reference system and a well known coordinate reference system given a set of pairs of corresponding coordinates. A site calibration enables the localization of new positions with cm accuracy using a RTK/PPK GNSS receiver and a transformation, which is a much more cost-effective than using a total station and much faster than other more traditional surveying methods. While site calibration is featured by almost every modern professional-grade GNSS receiver, the implementation is closed source and the output stored in proprietary file formats, tying the user to the vendor’s ecosystem. In this paper we propose a complete solution to the site calibration problem that can be fully implemented with open source software (in particular PROJ for coordinate transformations) and whose output can be represented in terms of an open standard (WKT version 2). Two methods and representations of a site calibration are described, a fully 3D one and a split horizontal and vertical one. Our main contribution is the openness and interoperability of the solution. Another important contribution is the analysis of the sensitivity of these solutions to measurement errors.

Content

- Introduction
- What is a Site Calibration?
  - Use case
  - Matching points
  - Mathematical concept
- WKT2: self contained CRS
- PROJ helps us

What is a Local Coordinate Reference System?

- Site CRS
- Local_CS
- Engineering_CRS
- Arbitrary

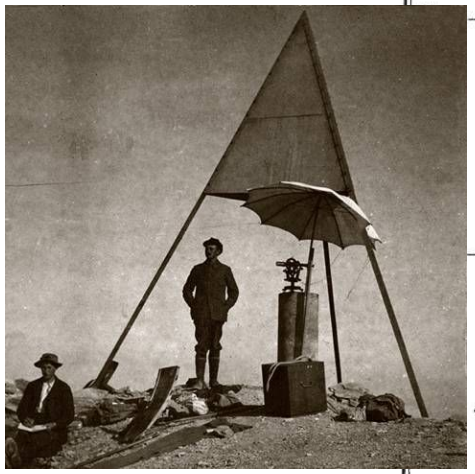


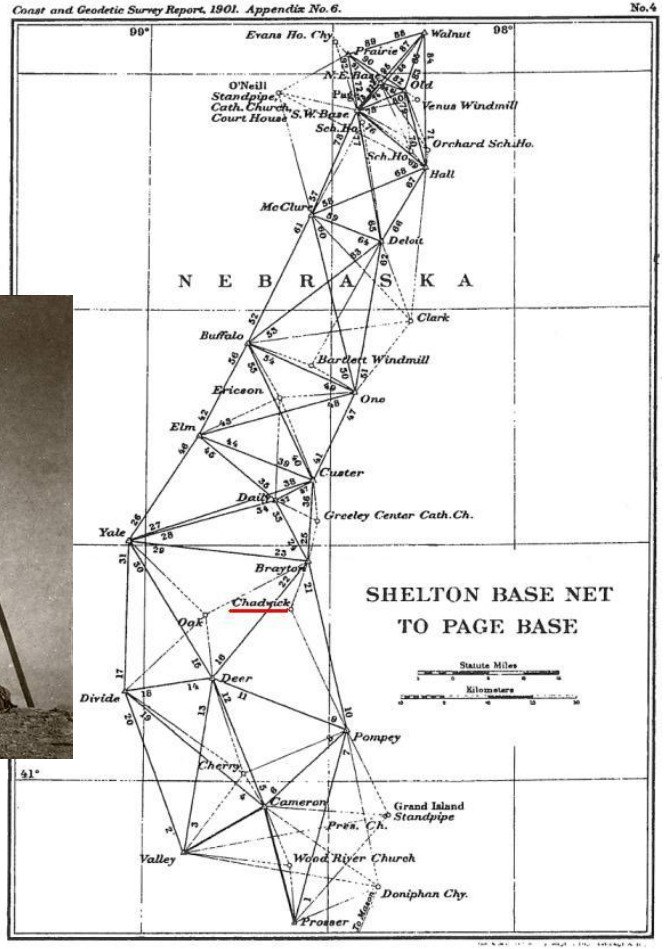
Cartesian Coordinate Reference System, **non-georeferenced**, with **arbitrary** origin, orientation and units.

Surveying without GNSS

Measuring angles and distances.

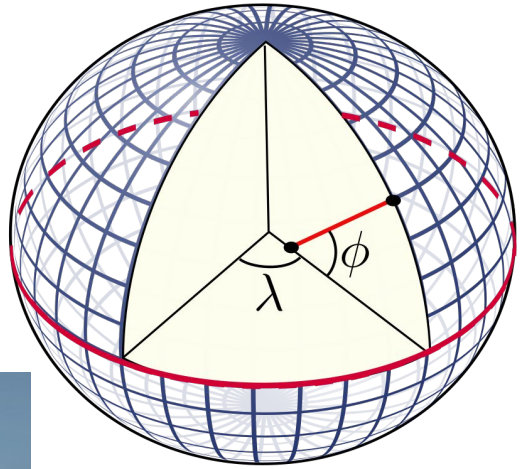
Triangle network.





Credit: Tom Cerchiara

Surveying with GNSS



Credit: Elmar Brokmann

Why do I need Site Calibration?

- Local CRSs are not geo-referenced to any global system.
- GNSS measurements are in global coordinates (official CRSs too).

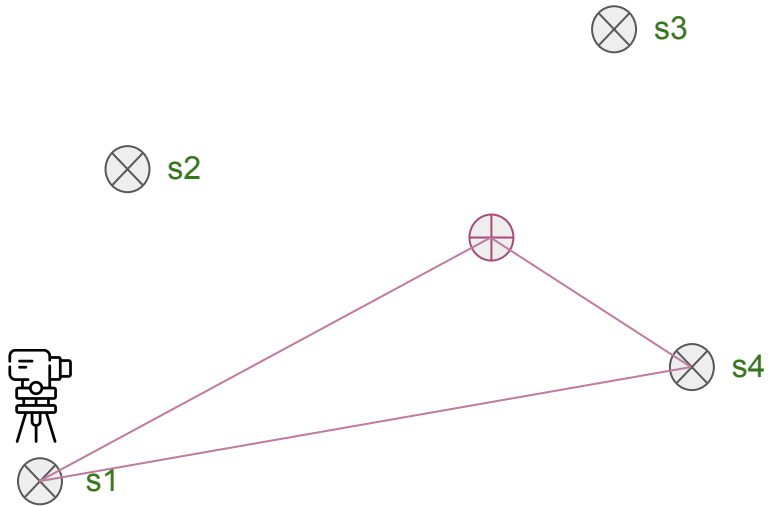
I would like to use my new GNSS device in my construction site…
How do we use a GNSS in a site system?

Site calibration: “How to” convert from *Local CRS* to the *Well Known CRS* (and vice versa)

Use case

- Now I can (quickly) measure with my GPS/RTK and transform to local coordinates (and vice versa).
- Fly my drone with a GPS, and produce output (point cloud, orthomosaic) in local coordinates.

Matching points (example)



Site coordinates (m)

s1	500.0	500.0	21.5
s2	512.5	546.2	22.3
s3	570.1	561.6	19.0
s4	574.6	506.3	26.4

Matching points (example)

g3 s3

g2 s2

g1 s1

g4 s4

Site coordinates (m)

s1	500.0	500.0	21.5
s2	512.5	546.2	22.3
s3	570.1	561.6	19.0
s4	574.6	506.3	26.4

Global coordinates (ETRS89 3D)
(lat, long, m)

g1	42.58624	6.23321	31.5
g2	42.66548	6.31015	32.2
g3	42.96531	6.80235	29.2
g4	43.19942	7.11174	36.4

* fictional points

What is missing?

Define the Site CRS in a “transformable” way. ✅

(Global CRS can be geographic or projected)

Site Calibrated CRS

- Base projected CRS
- Translation
- Rotation
- Scale

Always project the geographic CRS

- First we need a projected CRS.
- Create one with low distortion:
  - Centered in the area of interest
  - Conformal (not mandatory, but nice)
- UTM has already some distortion, avoid it.

Rotate, scale, translate

$$\mathbf{l} = c\mathbf{R}\mathbf{k}' + \mathbf{t}$$

$\mathbf{l}$: local coordinate
$\mathbf{k}'$: well know projected coordinate
c: scale

$\mathbf{R}$: rotation
$\mathbf{t}$: translation

- Rotate, scale, translate. Can be used with PROJ [Affine transformation]

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}^{dst} = \begin{bmatrix} xoff \\ yoff \\ zoff \end{bmatrix} + \begin{bmatrix} s11 & s12 & s13 \\ s21 & s22 & s23 \\ s31 & s32 & s33 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}^{src}$$

Solving it

Calculate the -rotation, scale, translation- that minimizes the residuals.

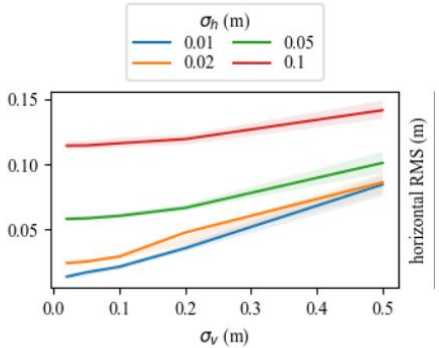
```
residual = (Local coord) - (transformed well known coord)
```

$$e^2(\mathbf{R}, \mathbf{t}, c) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{l}_i - (c\mathbf{R}\mathbf{k}'_i + \mathbf{t})\|^2$$

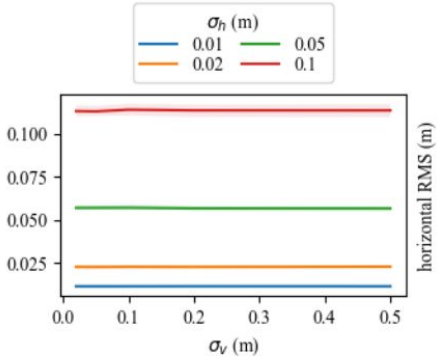
- Umeyama algorithm minimizes the residuals
- Works in n-dimesion (2 or 3 in our case)
- C++ Eigen library: `Eigen::umeyama`

Split horizontal and vertical calibration

- Accuracy of vertical and horizontal measurements are different
- Curvature of the earth noticed much more in elevations
- Use different points directly in the field
- Allows including a geoid model



a) 3D



b) 2D + 1D

Vertical calibration

Derived Vertical CRS using “vertical offset and slope” deriving conversion

Solved with Least Squares method

$$z' = z + Z_{off} + I_{\varphi} \cdot \rho_0(\varphi - \varphi_0) + I_{\lambda} \cdot \nu_0(\lambda - \lambda_0)\cos\varphi$$

Z_{off} : vertical offset

I_{φ}, I_{λ} : slopes in latitude and longitude

z, z' : elevation

λ, φ : latitude and longitude

What is WKT2?

“Well-known text representation of coordinate reference systems” (wikipedia)

We want to define the local CRS in an easy and **self contained** way.

Credit: Elmar Brokmann

WKT2 - projected centered in the area of interest

```
COMPOUNDCRS["Site Calibrated + Derived vertCRS",
    DERIVEDPROJCRS["Site Calibrated",
        BASEPROJCRS["Transverse Mercator centered in area of interest",
            BASEGEOGCRS["NAD83(2011)", ... ], // removed for clarity
            CONVERSION["Transverse Mercator",
                METHOD["Transverse Mercator",
                    ID["EPSG",9807]],
                PARAMETER["Latitude of natural origin",41.2305352787143,
                    ANGLEUNIT["degree",0.0174532925199433],ID["EPSG",8801]],
                PARAMETER["Longitude of natural origin",-73.1815861874286,
                    ANGLEUNIT["degree",0.0174532925199433],ID["EPSG",8802]],
                PARAMETER["Scale factor at natural origin",1,
                    SCALEUNIT["unity",1],ID["EPSG",8805]],
                PARAMETER["False easting",0,
                    LENGTHUNIT["metre",1],ID["EPSG",8806]],
                PARAMETER["False northing",0,
                    LENGTHUNIT["metre",1],ID["EPSG",8807]]]],
                ...
```

WKT2 - DerivingConversion (affine transformation)

```
…
DERIVINGCONVERSION["Affine transformation as PROJ-based",
    METHOD["PROJ-based operation method: +proj=pipeline
      +step +proj=affine +xoff=265262.95287
      +yoff=196619.27389 +s11=1.00003994119
      +s12=0.00548156923529 +s21=-0.00548156923529
      +s22=1.00003994119"]],
CS[Cartesian,2],
    AXIS["site east (x)",east,
        ORDER[1],
        LENGTHUNIT["metre",1,ID["EPSG",9001]]],
    AXIS["site north (y)",north,
        ORDER[2],
        LENGTHUNIT["metre",1,ID["EPSG",9001]]]],
…
```

WKT2 - VCRS + Vertical Offset and Slope

```
...
VERTCRS["Derived vertCRS",
    BASEVERTCRS["Ellipsoid (metre)",
        VDATUM["Ellipsoid"]],
    DERIVINGCONVERSION["Conv Vertical Offset and Slope",
        METHOD["Vertical Offset and Slope",
            ID["EPSG",1046]],
        PARAMETER["Ordinate 1 of evaluation point",41.2305352787143,
            ANGLEUNIT["degree",0.0174532925199433],ID["EPSG",8617]],
        PARAMETER["Ordinate 2 of evaluation point",-73.1815861874286,
            ANGLEUNIT["degree",0.0174532925199433],ID["EPSG",8618]],
        PARAMETER["Vertical Offset",31.0121985701957,
            LENGTHUNIT["metre",1],ID["EPSG",8603]],
        PARAMETER["Inclination in latitude",-6.12572852418232,
            ANGLEUNIT["arc-second",4.84813681109536E-06],ID["EPSG",8730]],
        PARAMETER["Inclination in longitude",-2.67487863214139,
            ANGLEUNIT["arc-second",4.84813681109536E-06],ID["EPSG",8731]],
        PARAMETER["EPSG code for Horizontal CRS",6318,
            ID["EPSG",1037]]],
    CS[vertical,1],AXIS["site up (z)",up,LENGTHUNIT["metre",1,ID["EPSG",9001]]]]]
```

PROJ

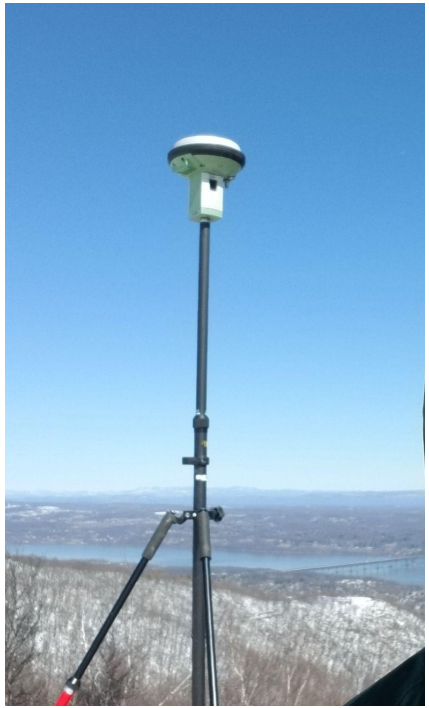
- PROJ (C/C++ API) we used to
  - Create initial CRSs
  - Produce a transverse mercator
  - Create Derived Projected, including **deriving conversion** method
  - Create Derived Vertical
  - Produce WKT2
  - Use output to transform coordinates

We would like to thank Even Rouault.

Main ideas

- Usage of WKT2 to define a CRS that encapsulates the site calibration.
- Self contained, plain text.
- Open source.
- Compound CRS
  - Derived Projected CRS
    - Centered to reduce distortion
    - Affine parametric transformation
  - Derived Vertical CRS
    - Vertical Offset and Slope

Thanks for watching!



Javier Jimenez Shaw

https://github.com/jjimenezshaw/