SITE CALIBRATION WITH PROJ AND WKT2

J. Jimenez Shaw *, J. Hernando, C. Strecha

Pix4D SA, Route de Renens 24, CH-1008 Prilly, Switzerland - (javier.shaw, juan.hernando, christoph.strecha)@pix4d.com

KEY WORDS: site localization, GNSS, WKT2, PROJ, coordinate transformations

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.

1. INTRODUCTION

In surveying projects in construction, civil engineering, mining, etc. it is common to work on coordinates referenced to a local coordinate reference system (CRS) that is established specifically for the project site. Taking construction as an example, plans for the object of interest are drawn in the local CRS, which needs to be established in the physical reality in order to be used to build as planned.

To establish a local CRS, the physical position of an initial control point is chosen and the bearing of the coordinate system axes is determined. Then, the entire project area is traversed extending the system point by point, measuring the coordinates of each control point relative to the previous one. These measurements are usually performed using electronic distance measurement (EDM) instruments with an integrated theodolite and levelling, more commonly known as total stations. A well calibrated total station can provide sub-millimeter accuracy for the control points.

The initial control points may be established before the design, but during project execution more detailed surveys are necessary for specific activities. The most appropriate method to use depends on the trade-off between required accuracy and cost. For example, laying out steel pillar requires higher accuracy than digging trenches for sewage. Locating coordinates or finding the coordinates of new positions can be achieved relatively fast and with high accuracy using a robotic total station. However, total stations are expensive equipment and need to be operated by skilled professionals, therefore using them for all surveying activities has a significant cost. More traditional methods require less expensive equipment, but at the expense of time and accuracy.

Geolocating positions in a national system can be performed with cm level accuracy by less skilled staff by means of a Global Navigation Satellite System (GNSS) receiver with real time kinematics (RTK) or post processing kinematics (PPK) (Fotopoulos and Cannon, 2001). This makes GNSS receivers a cost-reducing and time-saving alternative to other surveying techniques when cm accuracy is sufficient. Even cheaper consumer-grade receivers can be used if less accuracy is acceptable. However, the coordinates still need to be referenced to the local CRS used in the project. Albeit the local CRS is in principle arbitrary in georeferencing terms, its control points need to be also geolocated in a well known national system (such as EPSG:2056 for Switzerland) for legal and practical reasons such as placing buildings in cadastral maps. The correspondence between well known coordinates and local coordinates allows to obtain a transformation between both systems, which in turn enables the use of GNSS devices for more detailed surveys with less accuracy requirements.

A site calibration (or site localisation) is the process of finding a homeomorphism $s(\mathbf{k}; C)$ between the spaces of coordinates of a well known CRS $\mathcal{K}$ and the site local system $\mathcal{L}$, constrained by the set of control points C, with a minimal error in the area of interest. The error is some metric of the difference between $\mathbf{l}_i$ and $\mathbf{l}_i{}' = s(\mathbf{k}_i; C)$ for every pair $(\mathbf{l}_i, \mathbf{k}_i) \in C | \mathbf{l}_i \in \mathcal{L} \wedge \mathbf{k}_i \in \mathcal{K}$. In combination with GNSS surveying, this function allows the location of new positions with cm accuracy at a fraction of the cost of other high-accuracy surveying methods.

Many surveying devices provide a site calibration feature, but the algorithms are proprietary and the output stored in closed file formats. This effectively ties the user to the vendor ecosystem. In this paper we present a complete and interoperable solution that can be implemented purely in terms of open source software and standards. While the mathematical formulation is a well known and solved problem, to the best of our knowledge, the novelty of our approach resides in its complete openness.

Our main contribution is the precise description of the workflow involved in obtaining the mathematical solution of the site calibration problem and its representation as a self-contained coordinate reference system. In this context, self-contained means

* Corresponding author

that the final description of a site calibration embeds a well known CRS definition and the complete description of the function $s(\mathbf{k}; C)$. We show how the mathematical solution can be implemented using the open source library Eigen (Eigen contributors et al., 2023) As for the representation, our method relies on the OGC 18-010r7 open standard representation format (OGC, 2019b), commonly known as WKT version 2.

A site calibration can be solved in different ways. Another important contribution of this paper is the comparison and accuracy analysis of two mathematical methods that result in two different WKT2 representations. The first one solves a single 3D problem whereas the second splits the calibration in a horizontal and vertical component. We discuss the merits and disadvantages of each approach in terms of self-explainability of the solution and sensitivity to different types of measuring errors, in particular in the vertical axis, where GNSS receivers are known to have less accuracy.

We have tested these representations and coordinate transformations using the open source programming library PROJ version 9.2.0 (PROJ contributors et al., 2023). The combination of WKT2 and PROJ allow for off-the-shelf interoperability for any application using them in an open and standard manner. Part of the work carried out in this research has been contributed to PROJ 9.2.0, as previous versions lacked required functionality or suffered from implementation issues.

2. RELATED WORK

Well Known Text (WKT) is a syntax to represent different entities in text form. It is composed by ASCII keys and values enclosed in `[]`. See one example in appendix 6. WKT can be used with different types of data (like geometries), not only coordinate reference systems. However, in this paper we use only the specification for CRS, in particular version 2 of WKT (OGC, 2019b). Whereas the version 1 of WKT is widely used in GIS software, specially via GDAL (GDAL/OGR contributors, 2023), its scope is limited. This work is supported by some features that are only available in WKT2, specifically the possibility to derive the definition of a projected or vertical CRS based on an existing one and adding some transformation on top of it.

The mathematical methods used to solve the calibration have already been studied by Umeyama (1991) and Sorkine-Hornung and Rabinovich (2017) among others. These methods find a rigid transformation with scale that fits one set of points into another one. In the case of a site calibration, the transformation happens between a projection of the well known coordinates and the site coordinates. It has been demonstrated that these algorithms minimize the quadratic error of the residuals.

3. SITE CALIBRATION WITH WKT2

In WKT2, the degrees of freedom provided by deriving transformations on top of 2D/3D projected and vertical system allow for the representation of a site calibration in multiple ways. Based on this, this paper proposes two calibration algorithms whose solution can be entirely represented as a WKT2 string:

- **3D calibration** A calibration that is done globally in three dimensions. In this case 7 degrees of freedom are necessary: a rotation (3), a translation (3) and a scale (1). The final WKT represents a derived projected system, were the base projected CRS is custom defined for projecting the coordinates in $\mathcal{K}$ to a convenient Cartesian system and then an affine transformation is applied on top to obtain coordinates in $\mathcal{L}$.
- **A split 2D + 1D calibration** A horizontal calibration as above but in 2D (4 degrees of freedom), combined with a vertical calibration represented as a derived vertical system with a base system and a scalar transformation applied on top (one offset and two slope angles). Both parts are merged in a compound CRS.

While the former is simpler to implement, the latter approach has several advantages: not all points in C need to be used for both the vertical and the horizontal calibrations, even dedicated measurements can be used for each part; the minimum number of points in 2D is smaller than in 3D, allowing a rough localisation on the scene with just 2 points; the output parameters of the calibration are more easily understandable and checked by the user.

The use of WKT2 as a representation format is particularly convenient because it is a text-based representation that is very easy to store, transmit and process and also human readable. But not only that, it is the cornerstone of the interoperability of the solution proposed. The site calibration WKT2 embeds the well known CRS definition and through the transformation, this base system provides a geolocation for the local CRS. Any coordinate transformation library that implements the WKT2 standard and the transformation methods used will be able to easily transform coordinates not only between the base well known and the local systems, but also from any other system for which there exists a transformation to the base system. When using RTK/PPK, the right well known CRS (e.g. the CRS of the NTRIP mountpoint when using RTK) must be used in order for transformations to/from another CRS to be meaningful.

4. IMPLEMENTATION

In this section we describe the two methods to find and represent $s(\mathbf{k}; C)$. For each method, we detail what transformation is used, how to find the transformation parameters and its encoding as a WKT2 string. The terminology found both in OGC (2019b) and OGC (2019a) will be used. For naming transformation parameters we follow the EPSG Geodetic Parameter Dataset (IOGP, 2023) when applicable. In our implementation, the software library of choice is PROJ 9.2.0.

The 2D horizontal and the 3D calibrations are very similar. The output will be a `DERIVEDPROJCRS` in WKT2 (OGC, 2019b, §14.4) that will let us select any desired projection and define a deriving conversion `DERIVINGCONVERSION` (OGC, 2019b, §14.2) with an affine transformation to translate, rotate and scale it. The vertical calibration can be done with a derived vertical CRS (OGC, 2019b, §14.5) with a deriving conversion. In the split algorithm, the final output is a `COMPOUNDCRS` (OGC, 2019b, §15) made of two derived projected system.

A complete workflow for the 3D method has been implemented and released in PIX4Dcatch, a mobile photogrammetry application for scanning and collecting data. In combination with the viDoc GNSS receiver, PIX4Dcatch allows the measurement of control points with RTK-level precision. Control points can

be measured in a well known CRS and combined with manually typed or imported local coordinates measured with a total station to solve a site calibration. The solution can then be exported as a WKT2 string, a representation that can be directly consumed by other tools of the Pix4D ecosystem, such as PIX4Dmatic and PIX4Dcloud / PIX4Dcloud Advanced. Thanks to the use of PROJ it is possible to do coordinate conversions involving a site calibration without having to implement any new logic. The same observations will be extensible to the 2D + 1D method once it is integrated in PIX4Dcatch in the near future: the tools that currently support the 3D calibration output will support the split calibration with no change at all.

4.1 3D calibration

The core idea is to find a rigid body transformation with a scaling (a translation, rotation and scaling) that minimises some metric of $\mathbf{l}_i - s(\mathbf{k}_i; C)$ by means of a least-squares (LS) formulation (Sorkine-Hornung and Rabinovich, 2017).

The space $\mathcal{L}$ is Cartesian[1], but $\mathcal{K}$ will almost always use geographic coordinates. Before solving the LS problem $\mathcal{K}$ is projected to $\mathcal{K}_P$ using a Transverse Mercator projection with the centroid of all $\mathbf{k}_i$ as its projection center and a *scale factor at natural origin* of 1. The projection becomes the `BASEPROJCRS` of the final `DERIVEDPROJCRS`.

The choice of a transverse Mercator projection is somewhat arbitrary, but produces good results as the area of a site is usually not big enough to be affected by the projection distortion. This projection also seems to be commonly found in proprietary solutions. Any other conformal projection centered in the area of interest should work fine.

Before applying the LS method, it must be ensured that all coordinates are in the same units and use a right-handed coordinate system. On the other hand, the axis order and unit used in the WKT of the output derived projected CRS projection must follow the convention of the local CRS.

From the LS method we estimate a translation a rotation and a scale. These parameters are integrated in the derived projected CRS as a deriving conversion with a method named *"PROJ-based operation method"* containing a PROJ transformation pipeline for an affine transformation[2] such as `"+proj=pipeline +step +proj=affine +xoff=... +s11=... ..."`.

The parameters of the *"PROJ-based operation method"* are in metres and right-handed axis order. This transformation is applied after the projection, so any necessary unit conversions from the actual projected CRS and to the local CRS need to be included in the PROJ pipeline definition. For a projected and local CRS in ft, it is necessary to prepend to the pipeline the step `"+step +proj=unitconvert +xy_in=m +xy_out=ft +z_in=m +z_out=ft"` and append the reserve step. PROJ will handle any eventual axis swap to the local CRS.

4.1.1 Numerical computation The transformation parameters between $\mathbf{k}'_i \in \mathcal{K}_P$ and $\mathbf{l}_i \in \mathcal{L}$ and are estimated using the method by Umeyama (1991) for least-squares estimation of transformation parameters between two point patterns.

Umeyama's solution minimizes the mean squared error

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

Where $\mathbf{R}$ is a pure rotation matrix, c is the scale and $\mathbf{t}$ is the translation..

The algorithm is based on the analysis of the covariance matrix $\sum_{xy} \in \mathbb{R}^{d \times d}$ of the input point sets where d is corresponding to the dimension (2 or 3 in our case).

For the 3D transformation, The vector $\mathbf{t}$ is used as the translation of the PROJ affine transformation and $c\mathbf{R}$ as the matrix.

The Eigen function `Eigen::umeyama` was used in the C++ implementation. This function takes a source and a destination matrix, both with dimensions (d, n) and returns a homogeneous transformation with the form:

$$T = \begin{bmatrix} c\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

4.1.2 CRS construction with PROJ The final WKT was generated by constructing a derived projected CRS using PROJ's C++ API. Before solving the LS problem a transverse Mercator projection was created centred in the area of interest with `operation::Conversion::createTransverseMercator`. This projection was used to project the well known coordinates before solving the LS problem. Once the parameters of the affine transformation were obtained, we create the final derived projected CRS using the CRS transverse Mercator CRS mentioned above and the coordinate conversion. To build the coordinate conversion we used `operation::OperationMethod` and `operation::Conversion` for the affine transformation and finally we created a `crs::DerivedProjectedCRS`.

4.2 Horizontal calibration

The horizontal calibration is essentially the same as the 3D calibration, but in 2D. A possible variation in this case is to replace the PROJ transformation used in the deriving conversion with one of the coordinate operations published in the EPSG registry (IOGP, 2023). These operations are *"Affine parametric transformation"* (EPSG:9264)[3] and *"Similarity transformation"* (EPSG:9261)[4].

4.3 Vertical calibration

The main idea of the vertical calibration is to use a derived vertical CRS (OGC, 2019b, §14.5). In WKT2 this is represented as a `VERTCRS` with a `BASEVERTCRS` and a `DERIVINGCONVERSION`. The deriving conversion can be one of *"Vertical Offset and Slope"* (EPSG:1046)[5] or *"Vertical Offset"* (EPSG:9616)[6]. We

[1] In a local CRS where vertical coordinates have been measured with proper levelling methods, the XY plane follows an equipotential surface. Over wide areas, this surface is curved, but at a small scale it is fair to assume that the XY plane is flat and tangential to the level surface.

[2] https://proj.org/operations/transformations/affine.html (2023-03-22)

[3] https://epsg.org/coord-operation-method_9624/Affine-parametric-transformation.html (2023-03-22)

[4] https://epsg.org/coord-operation-method_9621/Similarity-transformation.html (2023-03-22)

[5] https://epsg.org/coord-operation-method_1046/Vertical-Offset-and-Slope.html (2023-03-22)

[6] https://epsg.org/coord-operation-method_9616/Vertical-Offset.html (2023-03-22)

will explain the method only for the former, as the latter is just a simplification. Note however that the estimation of the parameters for a vertical and slope transformation requires at least 3 points.

Strictly speaking, the current specification of WKT2 does not contain any provisions to represent a compound CRS where the vertical system represents ellipsoidal heights. The rationale is that a vertical system should be defined on its own, and the ellipsoidal heights are necessarily coupled to a horizontal definition that would be external. Furthermore, in WTK2 the notion of 3D systems was added, which questions the need for such a vertical system.

This limitation is not an issue if $\mathcal{K}$ is a compound CRS with an available geoid for the vertical part, but it prevents the usage of ellipsoidal heights for the base system of the derived vertical CRS. As a workaround we use the system `BASEVERTCRS[ "Ellipsoid (metre)", VDATUM["Ellipsoid"]]`, which is understood by PROJ to support 3D systems in LAS 1.4.

4.3.1 Numerical computation The formula of a vertical offset and slope transformation is described in IOGP (2019) §4.10.3:

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

where:
- Z_{off}: the offset of the transformation,
- I_φ: the value in radians of the slope parameter in the latitude domain,
- I_λ: the value in radians of the slope parameter in the longitude domain,

are variables, and:
- (φ, λ, z): the latitude, longitude and height of a point in $\mathcal{K}$,
- z': the target height in the local CRS,
- φ_0, λ_0: the latitude and longitude where the transformation is centered (e.g. one of the control points),
- ρ_0: the radius of curvature of the meridian at latitude φ_0, where $\rho_0 = a(1-e^2)/(1-e^2\sin^2\varphi_0)^{3/2}$,
- ν_o: the radius of curvature on the prime vertical at latitude φ_0, where $\nu_0 = a/(1-e^2\sin^2\varphi_0)^{1/2}$,
- a: the semi-major axis of the ellipsoid,
- e: the eccentricity of the ellipsoid,

are constants and latiude and longiude are angles in radians.

Finding Z_{off}, I_φ, I_λ for a set of control points C is equivalent to solve the linear system:

$$A \cdot \mathbf{x} = \mathbf{b}$$

where for every pair $((x_i, y_i, z_i'), (\varphi_i, \lambda_i, z_i)) \in C$:

$$A_i = \begin{bmatrix} 1 & \rho_0(\varphi_i - \varphi_0) & \nu_0(\lambda_i - \lambda_0)\cos\lambda_i \end{bmatrix}$$

$$b_i = \begin{bmatrix} z_i' - z_i \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} Z_{off} \\ I_\varphi \\ I_\lambda \end{bmatrix}$$

Our solver was written in C++ and uses the Bidiagonal Divide and Conquer SVD solver from Eigen (Eigen contributors et al., 2023).

```
Eigen::BDCSVD<Eigen::MatrixXd> bdcsvd(A);
Eigen::MatrixXd x = bdcsvd.solve(b);
```

5. RESULTS

In this section we provide some results of the algorihtms using synthetic and real data. Using synthetic gives full control of the noise present in the GNSS measurements, which allows to perform a sensitivity analysis. The results with real data are presented as as example.

5.1 Sensitivity analysis

The sensitivity to GNSS errors of algorithms has been analyzed using a large collection of synthetic data sets that have been crafted to mimic realistic site layouts.

Each layout has been created starting with a number of Poisson-disk samples N distributed on a horizontal unit-area square. The implementation from `scipy` version 1.10.1 has been used for Poisson-disk sampling. For assigning horizontal site coordinates, the square is scaled using some fixed size length l and randomly rotated around the Z axis. The heights are assigned independently by tilting the plane by t degrees (this tilting does not shrink the horizontal projection of the points) and adding a constant offset h_o and a zero-centered normally distributed variation with $\sigma = 3m$.

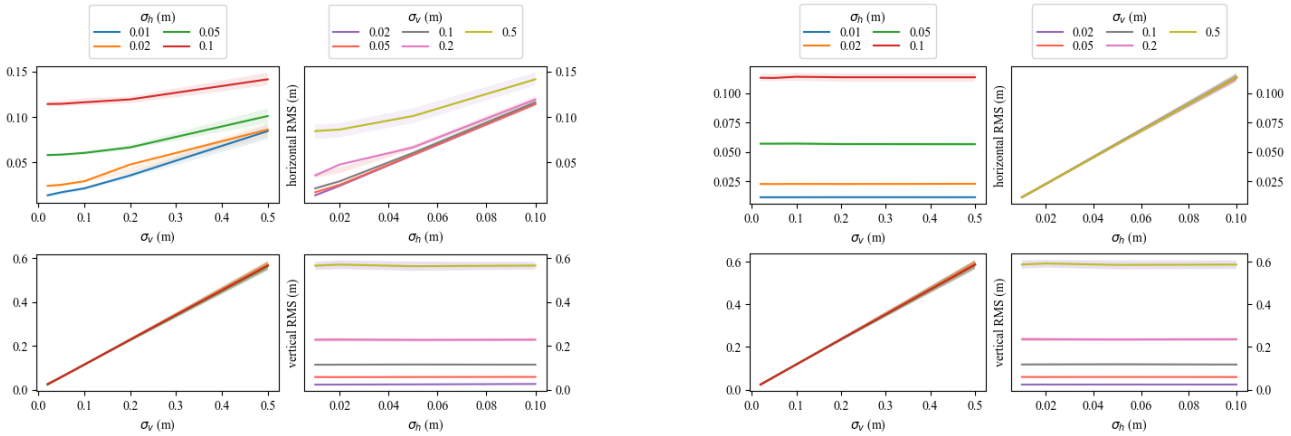
The site layout provides the set of local coordinates $C_\mathcal{L}$. In order to obtain the set of well known coordinates $C_\mathcal{K}$we define a transverse Mercator projection over a geographic CRS (which one is not important) at a random projection center and take $h/R+1$ as the scale factor, where R is the ellipsoid semi-major axis and h is the average height of $C_\mathcal{L}$. In order to account for the fact that a level surface is normally not tangent to the ellipsoid, the local heights are considered orthometric using EGM 2008 as the geoid model. This will need to be accounted for with some degree of rotation in the calibration.

Before transforming from $C_\mathcal{L}$to $C_\mathcal{K}$, some random noise is added separately to the horizontal and vertical parts using two zero-centered normal distributions with standard deviations σ_h and σ_v respectively. The horizontal noise is added as an XY shift in a uniformly random direction. The noisy local coordinates are finally unprojected from the compound system defined above (custom TM + EGM 2008 height) to geographic coordinates.

The parameter space used in the experiments is the Cartesian product of selected values in these ranges: $\sigma_h \in [0.01m, 0.1m]$, $\sigma_v \in [0.02m, 0.5m]$, $h_o \in [0m, 4000m]$, $t \in [1°, 20°]$, $N \in [4, 10]$ and $l \in [100m, 200m]$. In this parameter set we account for the fact that the geometric vertical dilution of precision (VDOP) of GNSS measurements is typically higher than the horizontal one (HDOP) (Spilker Jr., 1996). A minimum number of 4 points has been used because with 3 points the error metric used is very sensitive to measurement errors. This supports the common notion that in practice a site calibration should use at least 4 points.

For each combination, 200 different layouts were generated and calibrated using each of the methods. The validation of each calibration was not done using residuals computed from C, the error metric typically found in commercial devices. Instead, we generate 2 noisy points $\mathbf{l}_i^*$ randomly distributed inside the convex hull of $C_\mathcal{L}$, unproject them to $\mathcal{K}$ to obtain $\mathbf{k}_i^*$, and then measure the horizontal RMS:

$$\sqrt{\sum |\mathbf{l}_i^{*(j)} - s(\mathbf{k}_i^{*(j)}; C)|_{XY}^2}$$



(a) 3D calibration

(b) 2D+1D calibration

Figure 1. Horizontal RMS and vertical RMS of all site layouts tested for different values of σ_h and σ_v with the 3D and separate horizontal and vertical site calibration algorihtms. The solid lines represent the mean and the transparent stripes the interquartilic ranges. It can be observed that the in the split case, the horizontal RMS is independent of σ_v and the vertical RMS is independent of σ_h. This is not the case of the 3D algorithm, where σ_v can significantly affect the horizontal RMS.

and vertical RMS:

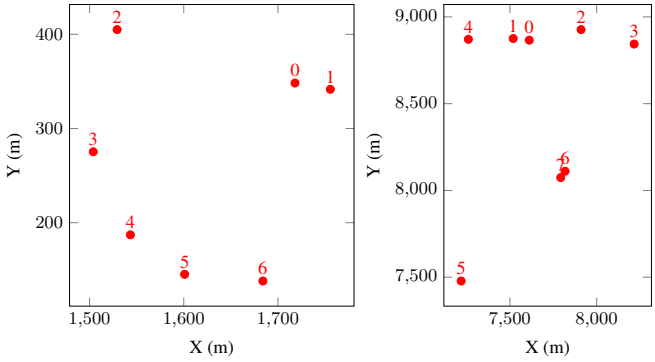
$$\sqrt{\sum ({\mathbf{l}_i^{*(j)}}_z - s(\mathbf{k}_i^{*(j)}; C)_z)^2}$$

for all $i \in \{1, 2\}$ and $j \in \{1..200\}$. For each pair (σ_h, σ_v) the means of the horizontal and vertical RMSs of the experiment series are computed.

Figures 1a and 1b contain plots of the relation between (σ_h, σ_v), the average horizontal RMS and the average vertical RMS for the 3D calibration and the split calibration respectively. The plots also show the inter-quartilic range of each variable as a transparent stripe of the same color as the plot line. The stripe only becomes clearly visible in a few cases, which means that the standard deviation of the RMS is low.

The first observation that can be made is that the horizontal and vertical RMS at each experiment is higher than the respective σ_h and σ_v of the input noise. This is expected, as the validation is performed with points that were not used for adjusting the transformation, and the residual errors will tend to be higher with the validation points than the control points, specially for experiments with a low number of samples.

As expected, in the 3D case the horizontal RMS is sensitive to noise in the vertical measurements. In real conditions the VDOP can be sometimes an order of magnitude higher than the HDOP, so this is an undesirable property. On the other hand, the split calibration does not suffer from this problem. This can be explained in terms of how the transformations operate. In the 3D case, the algorithm embeds the rotation angle around the Z axes and the 2 other angles for the vertical deflection introduced by the geoid in a single affine transformation. If the measured heights contain error, this error is propagated to the estimation of the slope angle and the affine transformation misplaces the horizontal coordinates, not only the vertical ones. The split transformation does not have this problem. Interestingly, the vertical RMS does not seem to be affected by σ_h. One possible explanation could be that since our terrains are relatively even (not horizontal, but flat), the inference of the vertical deflection is relatively insensitive to errors in the horizontal position. However, we would need to perform more experiments to validate this hypothesis.



(a) Points in dataset A

(b) Points in dataset B

Figure 2. Horizontal layout of the control points from dataset A and B, each one in its own local CRS (in m).

5.2 Real-world examples

The split calibration algorithm has been further validated with several real-world data sets. In this section we present two with 7 and 8 points for illustrative purposes. Dataset A has 7 3D point measurements as shown in figure 2a. The horizontal diameter of the dataset is about 309 m and the maximum vertical difference is 15.7 m. Dataset B has 8 measurements and is shown in figure 2b. The horizontal diameter of the dataset is about 1690 m and the maximum vertical difference is 7.8 m.

For each data set a split site calibration was solved and the residuals of each control point $(\mathbf{l}_i, \mathbf{k}_i) \in C$ were computed as $\mathbf{r}_i = \mathbf{l}_i - s(\mathbf{k}_i; C)$. In general, for GNSS-based land surveying acceptable residuals are within 1-2 cm, which is the around the accuracy of RTK/PPK measurements.

A leave-one-out cross-validation (LOOCV) has been carried out. We extract one point $\mathbf{c}_i \in C$ to be used as check point and solve the site calibration with the rest $C^i = C \backslash \{\mathbf{c}_i\}$. This process is repeated for each point and then we compute the RMS of the residuals as $\sqrt{\sum_i (\mathbf{l}_i - s(\mathbf{k}_i; C^i)^2}$.

The results for dataset A are show in table 1 and for dataset B in table 2. In both cases the residuals and LOOCV RMS are within acceptable values.

	X (m)	Y (m)	Z (m)
residuals	-0.00934	0.00145	-0.00604
	-0.00381	0.00252	-0.00374
	0.00044	-0.00209	0.00933
	0.00602	0.00922	-0.00164
	-0.00268	0.00296	-0.00259
	0.00399	0.00450	-0.01513
	0.00538	-0.01854	0.01981
LOOCV-RMS	0.00720	0.01149	0.01930

Table 1. Residuals and RMS of the LOOCV for dataset A in m.

	X (m)	Y (m)	Z (m)
residuals	0.00055	-0.00064	0.00608
	-0.00178	-0.01034	-0.00272
	0.00755	-0.00955	0.00340
	-0.00993	0.00939	-0.00214
	-0.00577	0.00672	-0.00352
	-0.01056	0.00377	0.00164
	0.01525	-0.00017	-0.00274
	0.00471	0.00077	-0.01683
LOOCV-RMS	0.01276	0.00881	0.00831

Table 2. Residuals and RMS of the LOOCV for dataset B in m.

6. CONCLUSIONS

In this paper we have presented two algorithms to solve the site calibration problem which can be implemented using the open source software libraries PROJ and Eigen and whose output can be represented in a self-contained WKT2 string, which is an open standard.

The first method is a fully 3D calibration. This method is easier to implement and its output is easier to represent in WKT. However, it is quite sensitive to vertical measurement error and it depends on a transformation method which is not registered in EPSG and might be difficult to integrate in any applications not using PROJ.

The second method is a split horizontal calibration. Although this solution is more elaborate, specially for representing correctly the output in a compound CRS with WKT2, it has two big advantages over the 3D one: the accuracy of the horizontal transformation is not sensitive to the errors of vertical measurements and it can be expressed using well-known transformations present in the EPSG registry.

The use of WKT2 makes it trivial to store and transmit the solution of a calibration. Any application that uses PROJ (9.2.0 or a more recent version) as its coordinate transformation library can consume site calibrations produced by other applications with virtually no change.

An alternative to our solution would be to define the system using an engineering CRS and define the transformation elsewhere. While this solution may appear to be more flexible, the advantages of having a single self-contained standard representation supported by PROJ become evident when coordinates need to be exchanged between applications.

It would be desirable to encode the scale, translation and rotation as part of the projected system parameterization, but no projected CRS whose implementation behaves robustly on any latitude was found in PROJ.

ACKNOWLEDGEMENTS

We would like to thank Even Rouault for the maintenance of the PROJ project and his prompt reaction to the issues that were reported during the development of this work.

REFERENCES

Eigen contributors, Jacob, B., Guennebaud, G., et al., 2023. Eigen. https://eigen.tuxfamily.org/. Visited on 2023-03-22.

Fotopoulos, G., Cannon, M. E., 2001. An overview of multi-reference station methods for cm-level positioning. *GPS solutions*, 4, 1-10.

GDAL/OGR contributors, 2023. GDAL/OGR Geospatial Data Abstraction software Library. doi.org/10.5281/zenodo.7764163.

IOGP, 2019. Geomatics Guidance Note Number 7, part 2, Coordinate Conversions and Transformations including Formulas. Technical Report 373-7-2, International Association of Oil and Gas Producers.

IOGP, 2023. EPSG Geodetic Parameter Dataset. https://epsg.org. Visited on 2023-03-22.

OGC, 2019a. OGC Abstract Specification Topic 2: Referencing by coordinates. Technical Report 18-005r4, Open Geospatial Consortium. Visited on 2023-03-22.

OGC, 2019b. Geographic information - Well-known text representation of coordinate reference systems. Technical Report 18-010r7, Open Geospatial Consortium. Visited on 2023-03-22.

PROJ contributors, Evenden, G. I., Rouault, E., Warmerdam, F., Evers, K., Knudsen, T., Butler, H., Taves, M., Schwehr, K., Sales de Andrade, E., Karney, C., Couwenberg, S., Dawson, N., Snow, A. D., Jimenez Shaw, J., 2023. PROJ coordinate transformation software library. doi.org/10.5281/zenodo.5884394.

Sorkine-Hornung, O., Rabinovich, M., 2017. Least-Squares Rigid Motion Using SVD. Technical report, ETH Zürich.

Spilker Jr., J. J., 1996. Satellite constellation and geometric dilution of precision. *Global Positioning System: Theory and Applications, Volume I*, American Institute of Aeronautics and Astronautics, Inc., 177–208.

Umeyama, S., 1991. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), 376-380. doi.org/10.1109/34.88573.

APPENDIX: WKT2 EXAMPLE

```
COMPOUNDCRS["Site Calibrated + Derived vertCRS",
 DERIVEDPROJCRS["Site Calibrated",
  BASEPROJCRS["Transverse Mercator centered in area of interest",
   BASEGEOGCRS["NAD83(2011)",
    DATUM["NAD83 (National Spatial Reference System 2011)",
```

```
    ELLIPSOID["GRS 1980",6378137,298.257222101,
     LENGTHUNIT["metre",1]]],
   PRIMEM["Greenwich",0,
    ANGLEUNIT["degree",0.0174532925199433]]],
  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]]],
  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]]]],
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]]]]
```


https://doi.org/10.5194/isprs-archives-XLVIII-4-W7-2023-75-2023