Scientific Research Publishing

Journal of Geographic Information System, 2022, 14, 430-443
https://www.scirp.org/journal/jgis
ISSN Online: 2151-1969
ISSN Print: 2151-1950

Georeferencing 3D Tiles Generated from Photogrammetry-Derived Mesh Using Ground Control Points

Kyongil Woo, Adrian Onsen, WonSok Kim

Institute of Remote Sensing and Geoinformatics, Academy of Science, Pyongyang, DPRK
Email: wugis1219@gmail.com

How to cite this paper: Woo, K., Onsen, A. and Kim, W. (2022) Georeferencing 3D Tiles Generated from Photogrammetry-Derived Mesh Using Ground Control Points. *Journal of Geographic Information System*, **14**, 430-443.
https://doi.org/10.4236/jgis.2022.145023

Received: September 4, 2022
Accepted: October 11, 2022
Published: October 14, 2022

Abstract

In order to achieve efficient visualization of massive 3D photogrammetry models on the web-based virtual globe, original photogrammetry 3D models should be converted into 3D Tiles with accurate georeferencing information to render on the correct surface position. There are a few software tools and online platforms for generating 3D Tiles and various ways to generate 3D Tiles were reported, but none of them do not provide a way to georeference 3D Tiles with GCPs. This paper is intended to suggest an approach to georeference 3D Tiles with GCPs based on the Kabsch algorithm which is very similar to georeferencing raster data with GCPs in the traditional GIS. First, we mathematically describe how to apply the Kabsch algorithm for georeferencing 3D Tiles with GCPs. Following this description, we demonstrate our implementation allowing constructing GCPs and calculating the "transform" property of 3D Tiles. Finally, we tested the suggested method in Construkted Reality which is the platform to stream and visualize massive photogrammetry and point clouds datasets.

Keywords

3D Tiles, GCP, Kabsch, 3D WebGIS, Cesium

1. Introduction

In recent years, photogrammetry technology has made tremendous progress technically and automatic 3D model reconstruction technique is becoming more and more mature. However, the significant amount of data associated with 3D models poses a challenge in terms of efficiently transmitting and visualizing 3D models on the Web-based 3D model viewer. To accommodate the streaming of

massive 3D models, various open 3D model format specifications such as 3D Tiles, Indexed 3D Scene Layer (I3S), and Spatial 3D Model (S3M) were developed, and among them, 3D Tiles exhibit greater flexibility, higher customizability, better portability, and openness [1].

3D Tiles [2] is an open specification for streaming and rendering massive multi-source heterogeneous 3D geospatial datasets. It defines a spatial data structure and a set of tile formats designed for 3D and optimized for streaming and rendering. Its primary purpose is to improve the streaming and rendering performance of massive heterogeneous datasets.

As 3D Tiles has gained widespread adoption as the preferred way for streaming massive 3D data on the web, more and more software platforms have added support for the 3D Tiles, and many photogrammetry software tools such as FME [3], Bently ContextCapture [4], Agisoft MetaShape [5], and RealityCapture [6] support exporting 3D Tiles.

In order to accurately render 3D Tiles in the real-world location, they should be georeferenced, and this can be done in various ways depending on how 3D Tiles are generated.

Many photogrammetry software tools use GCPs to ensure that a 3D model's reconstruction has the correct scale, orientation, or absolute position information [7] but, the way to georeference 3D Tiles generated from the 3D model with used GCPs which is very desirable has not yet been reported.

Therefore, in this paper, we suggest and implement the method to georeference 3D Tiles with GCPs based on the Kabsch algorithm [8], which is very similar to georeferencing raster data with GCPs in traditional GIS.

The remainder of the paper is organized as follows: Section 2 provides the review of related works, and the suggested idea is presented in Section 3, in Section 4 we demonstrate the implementation of our system. In Section 5 the validity is tested through experiments for sample 3D Tiles. Finally, Section 6 concludes the paper.

2. Related Works

Converting different kinds of geospatial data such as photogrammetry-lidar derived mesh, BIM, and point cloud into 3D Tiles has drawn much attention since the strength of 3D Tiles was known, and a few commercial software tools and platforms like Cesium ion [9], FME, Bently Context Capture, RealityCapture for it were reported. However, there are some limitations in customizing 3D Tiles generation because the conversion workflow is still a blackbox for users. Moreover, using them is infeasible in case the data are sensitive to the restricted public access requirement.

It is generally agreed that so far, no standard, a wholesome open-source solution exists for converting geospatial data into 3D Tiles. Therefore, many researchers have studied different tiling methods according to the characteristics of their data and the purpose of their project.

Schilling *et al.* [10] described the data processing steps from CityGML to 3D

Tiles, which used CESIUM_RTC [11] extension to georeference glTF. Gan *et al.* [12] suggested a method of generating 3D Tiles from DSM (digital surface model). Song *et al.* [13] proposed 3D Tiles dynamic loading and scheduling strategy based on terminal memory size and screen space error but did not explain how 3D Tiles used in their experimental was generated. Chen *et al.* [14] presented the workflow to convert IFC to 3D Tiles using multiple open source tools and libraries. Visuri *et al.* [15] used Cesium ion to convert their CityGML data to 3D Tiles. Kolarić *et al.* [16] implemented a set of routines for generating 3D Tiles that contain buildings with OpenStreetMap (OSM) data. Xu *et al.* [17] studied creating 3D Tiles from IFC using obj2gltf, and discussed how to calculate the transition matrix between the local coordinate system of IFC and EPSG:4978. Lu *et al.* [18] described the converting steps for generating 3D Tiles from WRCD (weather radar composing data) and they calculate a transform property of tileset.json to georeference 3D Tiles using the geographic coordinates of the WRCD center point. Mao *et al.* [19] proposed a workflow for generating 3D Tiles from CityGML. Li *et al.* [20] used objTo3dtiles to convert 3D electrical model to 3D Tiles but did not describe how their 3D Tiles was georeferenced. Jaillot *et al.* [21] suggested the temporal extension of 3D Tiles, and converted CityGML data of a 3DCityDB defined in a projected coordinate system to 3D Tiles using their own software, Py3DTiles for delivering time-evolving 3D city models on the web. Zhan *et al.* [1] presented a tiling algorithm of complex BIM models based on the R-tree.

In summary, although there were a few studies on generating 3D Tiles, to the best of our knowledge, no researchers have attempted to use GCPs to georeference 3D Tiles.

One of the easy and convenient ways to tile photogrammetry-derived 3D mesh is to use Cesium ion, which provides the "3D Tiles Location Editor" to georeference generated 3D Tiles. However, the "3D Tiles Location Editor" has a limitation in using because it is not easy to determine parameters like longitude, latitude, height, heading, pitch, and roll for georeferencing [22].

In the following section, technical specifications about our idea behind the georeferencing of 3D Tiles with GCPs are described.

3. Suggested Approach

We suggest a way to georeference 3D Tiles generated from a photogrammetry-derived mesh with GCPs using the Kabsch algorithm [8]. We do not mind how a mesh is converted to 3D Tiles and focus on only georeferencing generated 3D Tiles. And we assume coordinates of all vertices of 3D Tiles are defined in the local coordinate system and the given tileset does not contain any nested tileset.

3D Tiles can have georeferencing information in two ways: directly define vertex positions in WGS84, Earth-centered, Earth-fixed coordinate system (ECEF or EPSG:4978 [23]) using RTC_CENTER [24]; to define vertex positions in the local coordinate system and to specify the "transform" property of each tile so that all vertex positions are transformed into EPSG:4978.

The basic idea is to set up points corresponding to the given GCPs on the 3D Tiles and find the best transformation by which transformed corresponding points match GCPs, and set the "transform" property using calculated transformation. This is formalized to the problem to find the optimal/best rotation and translation between two sets of corresponding 3D points data (Figure 1).

To solve this problem, we use the Kabsch algorithm [8] which is the method for calculating the optimal translation and rotation that minimizes the Root Mean Squared Deviation (RMSD) between two paired sets of points. It is widely used in structural bioinformatics, molecular simulations, molecular modelling, structural biology research to superimpose pairs of molecules and physics simulation. Figure 2 shows the flowchart of Kabsch algorithm, and it is efficient and easy to implement because it only requires a few matrix operations.

However, the algorithm requires a predefined mapping between the sets of points, for example, in our case, the mapping of GCPs and corresponding points. We will describe how to setup such mapping in our system in Section 4.

Let $B_i, L_i, H_i \left(i = 1, 2, \cdots, n\right)$ be latitude, longitude, and height of GCPs and $p_i \left(i = 1, 2, \cdots, n\right)$ a point corresponding to ith GCP on the 3D Tiles.

So, $\boldsymbol{P}$ of the Kabsch algorithm can be directly constructed with p_i

$$\boldsymbol{P} = \begin{pmatrix} p_{x1} & p_{y1} & p_{z1} \\ p_{x2} & p_{y2} & p_{z2} \\ \vdots & \vdots & \vdots \\ p_{xn} & p_{yn} & p_{zn} \end{pmatrix} \tag{1}$$

$\boldsymbol{Q}$ of the Kabsch algorithm can be constructed by converting the geographic coordinate of GCPs into ECEF coordinate system.

$$\boldsymbol{Q} = \begin{pmatrix} q_{x1} & q_{y1} & q_{z1} \\ q_{x2} & q_{y2} & q_{z2} \\ \vdots & \vdots & \vdots \\ q_{xn} & q_{yn} & q_{zn} \end{pmatrix} \tag{2}$$

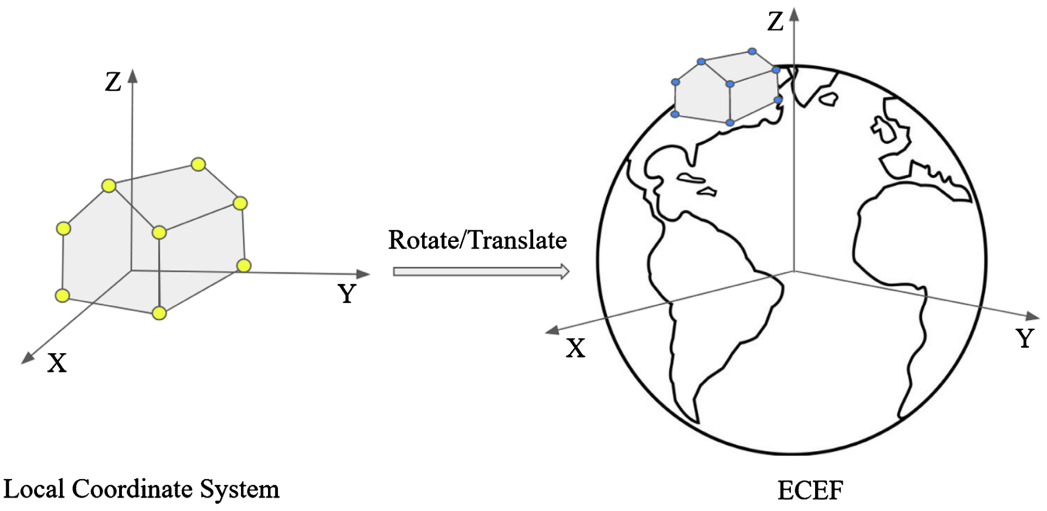

Figure 1. Georeferencing 3D Tiles by transformation.

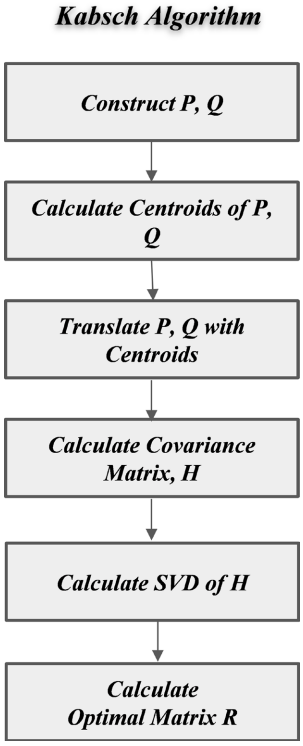

Figure 2. Flowchart of Kabsch algorithm.

where $\left(q_{xi}, q_{yi}, q_{zi}\right)$ is a cartesian coordinate in ECEF for $B_i, L_i, H_i - H_{\text{offset}} \left(i = 1, 2, \cdots, n\right)$. H_{offset} is a constant called "altitudeOffset", and its meaning is explained in the following section.

Let $\boldsymbol{C}_p, \boldsymbol{C}_q$ are centroids of GCPs and corresponding points.

$$\boldsymbol{C}_p = \frac{1}{n}\sum_{i=1}^{n}\left(p_{xi}, p_{yi}, p_{zi}\right) \tag{3}$$

$$\boldsymbol{C}_q = \frac{1}{n}\sum_{i=1}^{n}\left(q_{xi}, q_{yi}, q_{zi}\right) \tag{4}$$

Then the Translation step of the Kabsch algorithm is done using $\boldsymbol{C}_p, \boldsymbol{C}_q$, and $\boldsymbol{P}$, $\boldsymbol{Q}$ are updated so that their centroid coincides with $\boldsymbol{C}_p, \boldsymbol{C}_q$.

$$\boldsymbol{P} = \begin{pmatrix} p_{x1} - C_{px} & p_{y1} - C_{py} & p_{z1} - C_{pz} \\ p_{x2} - C_{px} & p_{y2} - C_{py} & p_{z2} - C_{pz} \\ \vdots & \vdots & \vdots \\ p_{xn} - C_{px} & p_{yn} - C_{py} & p_{zn} - C_{pz} \end{pmatrix} \tag{5}$$

$$\boldsymbol{Q} = \begin{pmatrix} q_{x1} - C_{qx} & q_{y1} - C_{qy} & q_{z1} - C_{qz} \\ q_{x2} - C_{qx} & q_{y2} - C_{qy} & q_{z2} - C_{qz} \\ \vdots & \vdots & \vdots \\ q_{xn} - C_{qx} & q_{yn} - C_{qy} & q_{zn} - C_{qz} \end{pmatrix} \tag{6}$$

Next, the optimal rotation matrix $\boldsymbol{R}$ that turns $\boldsymbol{P}$ as close as possible to $\boldsymbol{Q}$ is calculated using Singular Value Decomposition [25] module as follows.

$$\boldsymbol{H} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{P} \tag{7}$$

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V} = SVD(\boldsymbol{H}) \tag{8}$$

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^{\mathrm{T}} \tag{9}$$

As a result, we get a 3 × 3 matrix $\boldsymbol{R}$ and then, we convert it to a 4 × 4 matrix:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_{00} & \boldsymbol{R}_{10} & \boldsymbol{R}_{20} & 0 \\ \boldsymbol{R}_{01} & \boldsymbol{R}_{11} & \boldsymbol{R}_{21} & 0 \\ \boldsymbol{R}_{02} & \boldsymbol{R}_{12} & \boldsymbol{R}_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

Finally, the transformation is determined:

$$\boldsymbol{M} = \boldsymbol{T}_q \boldsymbol{R} \boldsymbol{T}_p \tag{11}$$

where,

$$\boldsymbol{T}_p = \begin{pmatrix} 1 & 0 & 0 & -C_{px} \\ 0 & 1 & 0 & -C_{py} \\ 0 & 0 & 1 & -C_{pz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

$$\boldsymbol{T}_q = \begin{pmatrix} 1 & 0 & 0 & C_{qx} \\ 0 & 1 & 0 & C_{qy} \\ 0 & 0 & 1 & C_{qz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}$$

Error for each GCP is estimated as a distance between the GCP and corresponding point transformed by the calculated $\boldsymbol{M}$. Based on our assumption above we can simply georeference 3D Tiles by setting the transform property of the root tile of the given 3D Tiles with $\boldsymbol{M}$.

4. System Demonstration

The suggested method in this paper has been adapted to the georeferencing module of Construkted Reality, which is the platform to stream and visualize massive photogrammetry and point cloud datasets with ease on the web. An online portal is available at https://construkted.com. The rendering and georeference module were developed with CesiumJS.

Once the photogrammetry-derived 3D mesh is uploaded, the backend of the Construkted Reality converts it to the 3D Tiles. The system detects whether or not the 3D Tiles are georeferenced and displays the 3D Tiles on the blank background if it does not have georeferencing information so that the user is notified that she/he needs to georeference 3D Tiles (**Figure 3**).

Then the user can start georeferencing on the "Multiple GCP Editor" panel (**Figure 4**), where it is possible to add necessary GCPs and input the latitude, longitude, and altitude of each GCP.

Next, by simply mouse clicking, users can specify points corresponding to all GCPs on the 3D Tiles (**Figure 5**).

The mapping info between GCPs and corresponding points on the 3D Tiles is stored in the "gcpData" metadata which is represented as a JSON object like

Figure 6.

Figure 3. Not georeferenced 3D Tiles.

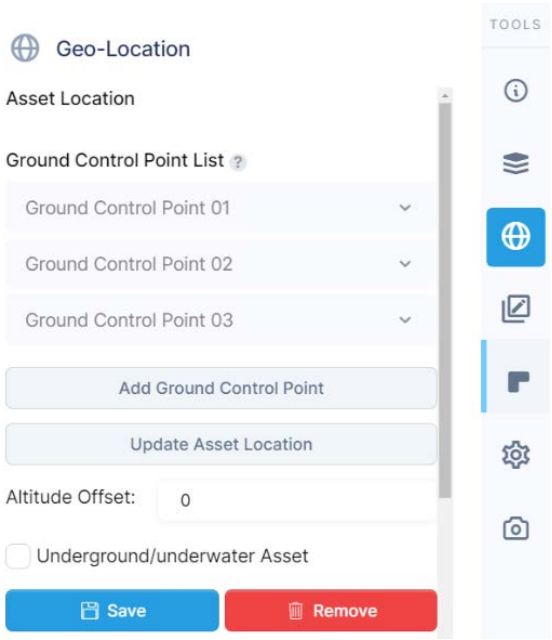

Figure 4. Multiple GCP Editor panel.

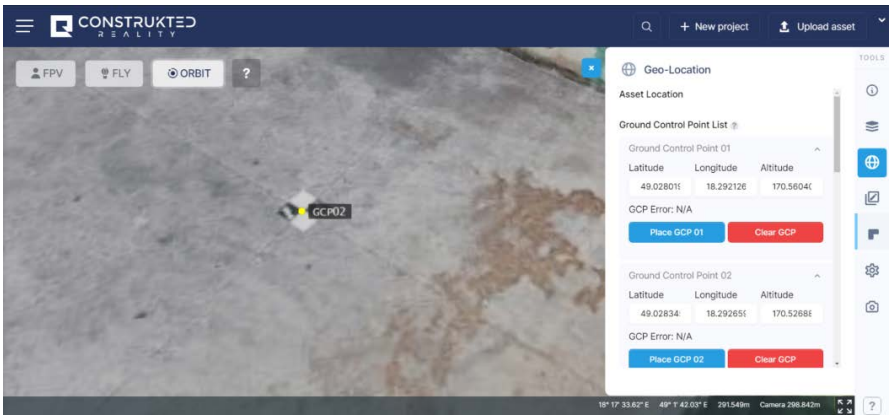

Figure 5. Specifying corresponding point.

```
{
  "gcps": [
    [18.29212697, 49.02801924, 170.560403],
    [18.29265927, 49.02834583, 170.526885],
    [18.29137971, 49.02883794, 169.964247],
    [18.28850684, 49.02991817, 171.202182],
    [18.29200705, 49.03060945, 172.10185],
    [18.2913444, 49.03018237, 170.784318],
    [18.29076568, 49.02984829, 170.337285],
    [18.29062673, 49.02954647, 170.09871],
    [18.28960265, 49.0303607, 171.012389],
    [18.29008487, 49.02921304, 170.217188],
    [18.29016613, 49.03086585, 171.712096],
    [18.28954766, 49.02921031, 170.561238],
    [18.290454, 49.03009902, 170.189142]
  ],
  "correspondingPoints": [
    [98.42027728330343, -168.46796852922313, 1.653520692139864],
    [137.30883767770447, -132.12997735745853, 1.644768643192947],
    [43.74755843055762, -77.38292201448543, 1.077072525396943],
    [-166.3300337750074, 42.757476830921455, 2.3016795599833 13],
    [89.65258267740039, 119.61285486161624, 3.21013806015253 07],
    [41.18476688821137, 72.12117490401164, 1.90258297231048 35],
    [-1.1522312108621389, 34.953853049680234, 1.458991950377822],
    [-11.320655068225115, 1.3982742659399914, 1.2050818176940084],
    [-86.21109659385485, 91.95394927108987, 2.1236795280128717],
    [-50.945483559348254, -35.69194431917913, 1.3263791799545288],
    [-45.003220379666544, 148.1308080423263, 2.8409833572804 93],
    [-90.23971631283632, -35.99115433250668, 1.691736487671733],
    [-23.94453130322913, 62.842433517632415, 1.30681190919131 04]
  ],
  "altitudeOffset": 130
}
```


Figure 6. JSON structure of "gcpData" metadata.

The "gcps" property in the "gcpData" metadata defines an array of ground control points defined in the geographic coordinates so that each element has the meaning of latitude, longitude, and altitude. The "correspondingPoints" property defines an array of cartesian points corresponding to ground control points. The "altitudeOffset" property is the offset value of altitude which is added to all ground control points' altitude in calculating georeferencing information. The surveyed altitude of GCP may be different than the altitude of the terrain on the virtual globe, so this is very useful in case users hope to correctly clamp 3D Tiles to the scene's terrain, for example, Cesium World Terrain [26].

At least 3 GCPs must be specified because the Kabsch algorithm requires a minimum of 3 unique points for a unique solution.

Once the user finishes the input of GCPs and corresponding points and clicks the "Update Asset Location" button, the system calculates $\boldsymbol{M}$ with GCPs and corresponding points as in described section 3, and displays an error for each GCP, and the camera of the scene flies to the calculated georeferenced location of the 3D Tiles (**Figure 7**).

5. Experiments

In order to evaluate the feasibility of the suggested method, several experiments were carried out with 3D models generated from sample datasets of OpenDro-

neMap [27], Pix4DMapper [28], and RealityCapture [6] that have GCPs (Appendix A).

5.1. Experiment 1

The dataset used in the experiment is an example dataset, "sheffield_cross" for OpenDroneMap, which has 5 GCPs defined in the WGS 84/UTM zone 17N. This dataset was processed through WebODM Lighting [29]. Points corresponding to GCPs were visually identified on the 3D Tiles using the GCP file and images included in the dataset (Figure 8).

Table 1 shows calculated errors for all GCPs, where Latitude, Longitude, and

Figure 7. The result of georeferencing.

Figure 8. Georeferenced "sheffield_cross".

Table 1. Calculated errors for GCPs of sheffield_cross.

Latitude	Longitude	Altitude	x	y	z	Error (m)
28.04109	−82.697388	3	−32.804	−12.103	2.883	0.5042
28.041439	−82.697094	3	−3.835	25.683	2.962	0.5104
28.041444	−82.696838	3	21.709	25.551	3.128	0.3332
28.041009	−82.696876	3	17.289	−22.478	3.172	0.2426
28.041064	−82.697071	3	−0.837	−15.971	3.241	0.8279

Altitude columns represent the geographic coordinates of a ground control point, and x, y, and z columns are the cartesian coordinate of a corresponding point defined in the 3D Tiles' local coordinate system.

5.2. Experiment 2

In this experiment, we used an example project, "quarry" for PIX4DMapper. GCPs are defined in EPSG:21781 [30] and corresponding points were visually identified using GCP overview images of the dataset and 3D GCP Markers of the project (**Figure 9**).

Figure 10 and **Table 2** show the result of georeferencing of 3D Tiles for this project.

5.3. Experiment 3

We used a sample dataset called "Drone Imagery + Ground control points" for RealityCapture, where all GCPs are defined in EPSG:4258 [31]. All GCPs are marked on the texture of the generated 3D model like a small section of a checkerboard (**Figure 11**) which leaves very little ambiguity about where the "point" of the ground control point is so we could identify all GCPs clearly on the texture of the 3D Tiles.

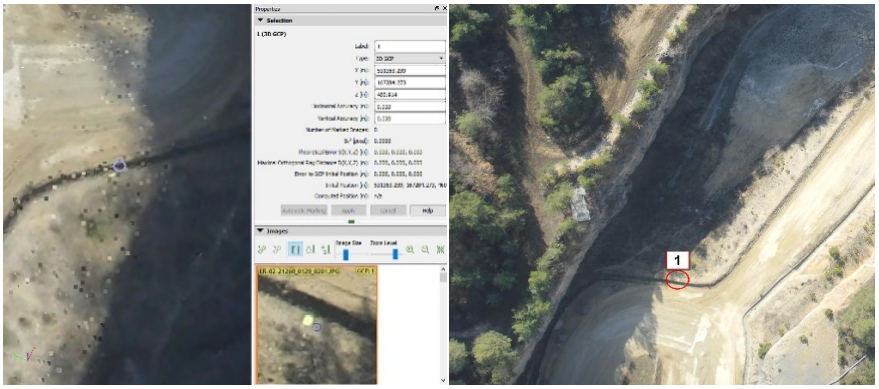

Figure 9. 3D GCP marker and overview image about first GCP.

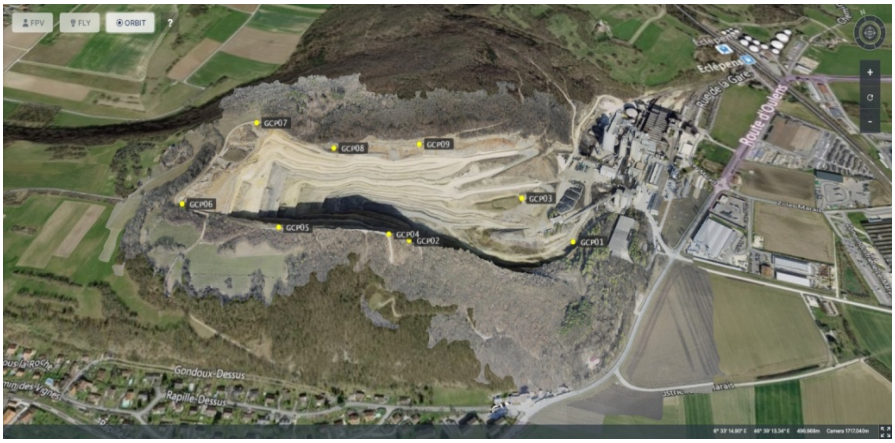

Figure 10. Georeferenced "quarry".

Figure 12 and Table 3 show the result of georeferencing 3D Tiles. As can be known in Table 3, the errors on all GCPs have a margin of a few centimeters.

Table 2. Calculated errors for GCPs of quarry.

Latitude	Longitude	Altitude	x	y	z	Error (m)
46.6532533	6.5418412	460.914	382.899	−224.233	−74.065	0.2424
46.6545008	6.5374464	458.742	48.011	−81.644	−76.223	0.2081
46.6546023	6.5411155	474.195	328.933	−73.466	−60.706	0.2721
46.653783	6.5365471	572.394	−20.742	−161.043	37.486	0.8062
46.6547443	6.5338956	566.361	−223.055	−51.687	31.372	0.2534
46.6560867	6.5315791	558.969	−398.627	99.748	24.064	0.4043
46.657881	6.534205	565.863	−196.142	296.976	30.860	0.4178
46.6563958	6.5360955	574.465	−53.048	130.127	39.491	0.1756
46.6558696	6.538516	574.874	131.609	69.446	39.927	0.1966

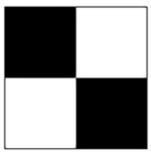

Figure 11. The shape of GCPs on the texture of the 3D model.

Figure 12. The result of georeferencing.

Table 3. Calculated errors for all GCPs.

Latitude	Longitude	Altitude	x	y	z	Error (m)
18.2921697	49.02801924	170.56	98.4202	−168.4679	1.6535	0.0027
18.29265927	49.02834583	170.52	137.3088	−132.1299	1.6447	0.0227
18.29137971	49.02883794	169.96	43.74755	−77.3829	1.0770	0.0218
18.28850684	49.02991817	171.20	−166.3300	42.7574	2.3016	0.0223
18.29200705	49.03060945	172.10	89.6525	119.6128	3.2101	0.0205

Continued

18.2913444	49.03018237	170.78	41.1847	72.1211	1.9025	0.0125
18.29076568	49.02984829	170.33	−1.1522	34.9538	1.4589	0.0117
18.29062673	49.02954647	170.09	−11.3206	1.3982	1.2050	0.0106
18.28960265	49.0303607	171.01	−86.2110	91.9539	2.1236	0.0105
18.29008487	49.02921304	170.21	−50.9454	−35.6919	1.3263	0.0092
18.29016613	49.03086585	171.71	−45.0032	148.1308	2.8409	0.0165
18.28954766	49.02921031	170.56	−90.2397	−35.9911	1.6917	0.0218
18.290454	49.03009902	170.18	−23.9445	62.8424	1.3068	0.0076

6. Conclusion

From experiments in this paper, it can be concluded that the Kabsch algorithm can be applied to georeferencing 3D Tiles using GCPs. Georeferencing errors were inevitably affected by the accuracy of "clicking on the 3D Tiles" for specifying points corresponding to GCPs. In experiment3 error was very small enough to accept the georeferencing result because it was easy to recognize GCP on the 3D Tiles. In order to clamp 3D Tiles to Cesium World Terrain, different values of "altitudeOffset" were used for the reason that Cesium World Terrain does not provide elevation relative to the WGS84 ellipsoid.

Data Availability Statement

Datasets used in this study are available at the following links.

https://github.com/pierotofy/drone_dataset_sheffield_cross/

https://support.pix4d.com/hc/en-us/articles/360000235126-Example-projects-real-photogrammetry-data#quarry

https://www.capturingreality.com/SampleDatasets

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zhan, W., Chen, Y. and Chen, J. (2021) 3D Tiles-Based High-Efficiency Visualization Method for Complex BIM Models on the Web. *ISPRS International Journal of Geo-Information*, **10**, Article No. 476. https://doi.org/10.3390/ijgi10070476

[2] 3D Tiles Format Specification. https://github.com/CesiumGS/3d-tiles/tree/main/specification#tile-format-specifications/

[3] FME Desktop. https://www.safe.com/fme/fme-desktop/

[4] ContextCapture Viewer. https://www.bentley.com/en/products/brands/contextcapture

[5] Agisoft MetaShape. https://www.agisoft.com/

[6] Reality Capture. https://www.capturingreality.com/

[7] PIX4D (2019, December 4) Ground Control Points: Why Are They Important? https://www.pix4d.com/blog/why-ground-control-points-important

[8] Wikipedia (n.d.) Kabsch Algorithm. https://en.m.wikipedia.org/wiki/Kabsch_algorithm

[9] Cesium ion. https://cesium.com/platform/cesium-ion/

[10] Schilling, A., Bolling, J. and Nagel, C. (2016) Using glTF for Streaming CityGML 3D City Models. *Proceedings of the* 21*st International Conference on Web3D Technology*, Anaheim, 22-24 July 2016, 109-116. https://doi.org/10.1145/2945292.2945312

[11] CESIUM_RTC. https://github.com/KhronosGroup/glTF/blob/main/extensions/1.0/Vendor/CESIUM_RTC/README.md

[12] Gan, L., Li, J. and Jing, N. (2017) Hybrid Organization and Visualization of the DSM Combined with 3D Building Model. *Proceedings of the* 2017 2*nd International Conference on Image*, *Vision and Computing*, Chengdu, 2-4 June 2017, 566-571. https://doi.org/10.1109/ICIVC.2017.7984619

[13] Song, Z. and Li, J. (2018) A Dynamic Tiles Loading and Scheduling Strategy for Massive Oblique Photogrammetry Models. *Proceedings of the* 2018 3*rd IEEE International Conference on Image*, *Vision and Computing*, Chongqing, 27-29 June 2018, 648-652. https://doi.org/10.1109/ICIVC.2018.8492731

[14] Chen, Y., Shooraj, E., Rajabifard, A. and Sabri, S. (2018) From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium. *ISPRS International Journal of Geo-Information*, **7**, Article No. 393. https://doi.org/10.3390/ijgi7100393

[15] Visuri, H., Jokela, J., Mesterton, N., Latvala, P., Aarnio, T. (2019) Producing and Visualizing a Country-Wide 3D Data Repository in Finland. *International Archives of the Photogrammetry*, *Remote Sensing and Spatial Information Sciences*, **XLII-4/W15**, 105-110. https://doi.org/10.5194/isprs-archives-XLII-4-W15-105-2019

[16] Kolarić, S. and Shelden, D. (2019) DBL SmartCity: An Open-Source IoT Platform for Managing Large BIM and 3D Geo-Referenced Datasets. *Proceedings of the* 52*nd Hawaii International Conference on System Sciences*, Hawaii, 8 January 2019, 1965-1974. https://doi.org/10.24251/HICSS.2019.238

[17] Xu, Z., Zhang, L., Li, H., Lin, Y. and Yin, S. (2020) Combining IFC and 3D Tiles to Create 3D Visualization for Building Information Modeling. *Automation in Construction*, **109**, Article ID: 102995. https://doi.org/10.1016/j.autcon.2019.102995

[18] Lu, M., Wang, X., Liu, X., Chen M, Bi, S., Zhang, Y. and Lao, T. (2020) Web-Based Real-Time Visualization of Large-Scale Weather Radar Data Using 3D Tiles. *Transactions in GIS*, **25**, 25-43. https://doi.org/10.1111/tgis.12638

[19] Mao, B., Ban, Y. and Laumert, B. (2020) Dynamic Online 3D Visualization Framework for Real-Time Energy Simulation Based on 3D Tiles. *ISPRS International Journal of Geo-Information*, **9**, Article No. 166. https://doi.org/10.3390/ijgi9030166

[20] Li, Y., Zhang, H. and Zhang, Q. (2021) A Framework for Interactive Online 3D Visualization of Electric Information. *Journal of Physics*: *Conference Series*, **1757**, Article ID: 012170. https://doi.org/10.1088/1742-6596/1757/1/012170

[21] Jaillot, V., Servigne, S. and Gesquière, G. (2020) Delivering Time-Evolving 3D City Models for Web Visualization. *International Journal of Geographical Information Science*, **34**, 2030-2052. https://doi.org/10.1080/13658816.2020.1749637

[22] Cesium (n.d.) Set Location for Data Uploaded to Cesium Ion. https://cesium.com/learn/3d-tiling/ion-tile-set-location/

[23] EPSG:4978. https://spatialreference.org/ref/epsg/wgs-84-2/

[24] 3D Tiles Reference Card. https://github.com/CesiumGS/3d-tiles/blob/main/3d-tiles-reference-card.pdf

[25] Wikipedia (n.d.) Singular Value Decomposition. https://en.wikipedia.org/wiki/Singular_value_decomposition

[26] Cesium World Terrain. https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/

[27] OpenDroneMap. https://www.opendronemap.org/

[28] PIX4Dmapper. https://www.pix4d.com/product/pix4dmapper-photogrammetry-software

[29] WebODM Lightning. https://webodm.net/

[30] EPSG:21781. https://spatialreference.org/ref/epsg/ch1903-lv03/

[31] EPSG:4258. https://spatialreference.org/ref/epsg/etrs89/

Appendix A

Results presented in this study can be found at the following links.

https://construkted.com/asset/a1fg3o0r69r/

https://construkted.com/asset/avajuri6r3g/

https://construkted.com/asset/avplbp844b3/