Technical Note

An AI-Based Workflow for Fast Registration of UAV-Produced 3D Point Clouds

Yong Feng [1,†], Ka Lun Leung [2,†], Yingkui Li [3,*] and Kwai Lam Wong [4]

1 Department of Computer Science, The City University of Hong Kong, Hong Kong, China; yongfeng2-c@my.cityu.edu.hk
2 Department of Mathematics, The Chinese University of Hong Kong, Hong Kong, China; 1155157403@link.cuhk.edu.hk
3 Department of Geography and Sustainability, University of Tennessee, Knoxville, TN 37996, USA
4 Department of Mechanical, Aerospace and Biomedical Engineering, University of Tennessee, Knoxville, TN 37996, USA; kwong@utk.edu
* Correspondence: yli32@utk.edu
† These authors contributed equally to this work.

Abstract: The integration of structure from motion (SFM) and unmanned aerial vehicle (UAV) technologies has allowed for the generation of very high-resolution three-dimensional (3D) point cloud data (up to millimeters) to detect and monitor surface changes. However, a bottleneck still exists in accurately and rapidly registering the point clouds at different times. The existing point cloud registration algorithms, such as the Iterative Closest Point (ICP) and the Fast Global Registration (FGR) method, were mainly developed for the registration of small and static point cloud data, and do not perform well when dealing with large point cloud data with potential changes over time. In particular, registering large data is computationally expensive, and the inclusion of changing objects reduces the accuracy of the registration. In this paper, we develop an AI-based workflow to ensure high-quality registration of the point clouds generated using UAV-collected photos. We first detect stable objects from the ortho-photo produced by the same set of UAV-collected photos to segment the point clouds of these objects. Registration is then performed only on the partial data with these stable objects. The application of this workflow using the UAV data collected from three erosion plots at the East Tennessee Research and Education Center indicates that our workflow outperforms the existing algorithms in both computational speed and accuracy. This AI-based workflow significantly improves computational efficiency and avoids the impact of changing objects for the registration of large point cloud data.

Keywords: point cloud; image registration; image segmentation; deep learning

check for updates

Citation: Feng, Y.; Leung, K.L.; Li, Y.; Wong, K.L. An AI-Based Workflow for Fast Registration of UAV-Produced 3D Point Clouds. *Remote Sens.* **2023**, *15*, 5163. https://doi.org/10.3390/rs15215163

Academic Editors: Riccardo Roncella and Danfeng Hong

Received: 13 September 2023
Revised: 16 October 2023
Accepted: 27 October 2023
Published: 29 October 2023

1. Introduction

The recent advances of structure from motion (SfM) and unmanned aerial vehicle (UAV), commonly known as drones, technologies have allowed the generation of up to millimeter and centimeter resolution digital surface models (DSMs) to quantify surface processes [1–3], to be used as a base map for geologic mapping [4], to derive vegetation structures [5], to investigate soil erosion [6], and to extract urban features [7,8]. SfM is developed in computer vision for three-dimensional (3D) object reconstruction based on highly overlapped photos taken with commonly used cameras, smartphones, and tablets [9,10]. SfM significantly reduces the cost of the land survey because the traditional photogrammetry requires specific and expensive cameras and aircrafts to reconstruct both natural and man-made land features. Specifically, the integration of SfM with low cost UAVs has the capability to conduct large-scale surface reconstructions for geographic and environmental studies [11–16].

A common SfM-UAV-based 3D topographic reconstruction includes two stages. The first stage is to capture a set of photos of the land by the cameras on UAVs. These photos are then used to generate a point cloud, a digital surface model of the mapped area, and an orthomosaic photo to provide a detailed and accurate representation of the land's topography. Several software programs, such as Pix4DMapper 4.6 and Agisoft Photoscan 1.4, can perform this task and their effectiveness has been proven in numerous studies [17–19].

The reconstructed surface can be further used to detect surface changes if multiple surveys are conducted over time. Such analysis requires the precise alignment, called registration, of the point clouds obtained by different surveys together. Several existing algorithms, such as the Iterative Closest Point (ICP) [20], Globally Optimal ICP (Go-ICP) [21], or Fast Global Registration (FGR) method [22], have been developed for point cloud registration. However, these algorithms are mainly developed for the registration of small and static point cloud data, but do not perform well when dealing with large point cloud datasets. In addition, some parts of the surface may be changed over time, and including the changing parts reduces the registration accuracy. Consequently, in practice, many studies still rely on the registration of the point clouds based on manually selected point pairs, which is time-consuming, labor-intensive, and subjective. Therefore, there is an urgent need for the development of a fast, accurate, and objective point cloud registration method.

In this paper, we propose an AI-based workflow to address the aforesaid problem based on the point cloud and ortho-photo generated by the UAV photos using SfM software, such as Pix4DMapper 4.6 and Photoscan 1.4. This workflow is implemented with Cloudcompare, a free and open source software (https://www.danielgm.net/cc/, accessed on 16 July 2023), for the point cloud registration. The innovation of this workflow is that we first implement an AI-based object-detection model to derive the bounding boxes of reference objects on the 2D ortho-photo, such as black/white targets, houses, and pipes. Then, we clip the point clouds of the selected reference objects based on the coordinates of the bounding boxes on the ortho-photo and only use these partial point clouds for the registration. This partial registration speeds up the registration process and avoids the potential impacts of moving objects on the registration. Finally, the transformation matrix derived from the partial registration is applied to the whole point cloud to complete the whole registration. We conduct experiments on the datasets that were collected for monitoring hillslope erosion on the erosion plots of the University of Tennessee. Detailed information on the plots is described in Section 4. Our major objective is to ensure the high-quality registration of the point clouds generated using UAV-collected photos. This proposed workflow can be used for other UAV-related applications as well, apart from soil-erosion measurement.

The rest of the paper is organized as follows. Section 2 provides relevant background studies and the motivation. Section 3 introduces our workflow and lists the algorithm and software we used. In Section 4, we present the experimental results conducted with our proposed workflow. In Section 5, we provide our analysis and discussion of the proposed workflow. Finally, we summarize the conclusions and discuss possible future development in Section 6. The codes used in our study can be found at https://github.com/hahaBlizzard/Main-Code.git (accessed on 16 October 2023).

2. Background and Motivation

Several algorithms have been developed for 3D point cloud registration. In 1992, Besl and his team proposed the ICP algorithm [20], which addresses the registration problem using a point-to-point approach. The ICP algorithm iteratively computes a registration vector through a least square quaternion operation until the error measure decreases to a low level. However, the ICP method has been criticized for its heavy computation [23], with a quadratic time complexity of $O(N_p N_x)$, where N_p and N_x represent the number of points in the two point cloud datasets. Additionally, the point cloud data need to be given a good initial rotation state in order to enhance the performance of the ICP method.

Since then, several variants of the ICP method have been developed to tackle the problem from a local approach [24,25]. However, this approach introduces the issue of local minima, which can degrade the overall performance of the algorithms. As a result, global registration methods have been explored. Yang et al. (2016) introduced the Go-ICP method [21], a global version of the ICP method that incorporates a nested Branch and Bound Algorithm as a global optimization step. Furthermore, Zhou et al. (2016) created the FGR algorithm, which also addresses the registration problem from a global perspective [22]. Their algorithm requires computing normals of points in a point set, which is essential for calculating the Fast Point Feature Histogram of the two point cloud datasets. Based on these histograms, an initial correspondence set is generated, and the global registration is achieved by solving an optimization problem.

The aforementioned algorithms have demonstrated their superiority over others in terms of efficiency and accuracy through some conducted experiments. However, most of those experiments have only dealt with point clouds consisting of an average of approximately 14,000 points. On the other hand, several million points are common to represent geographic data. Registering the entire point cloud set using any of the above algorithms takes a significant amount of time, and the resulting accuracy is often unsatisfactory because of the inclusion of changing parts, noises, different point densities, and so on. Our proposed workflow focuses mainly on improving the registration efficiency and accuracy of large-scale geographic data.

3. Proposed Workflow

The registration process is mathematically a coordinate transformation based on a transformation matrix, representing offset, rotation, and scale changes. Let P be the point cloud to be aligned and Q be the reference point cloud. Let P_C and Q_C be the point cloud generated from reference objects chosen from P and Q, respectively. There are several requirements for selecting the reference object:

1. the reference object must be fixed in position;
2. the area enclosed by the reference object should include the target object for registration.
3. The appearance and location of the reference object will not change over time.

Let P_T and Q_T be the target objects of P and Q, respectively. Suppose a registration is performed on P_C to align it to Q_C and the resultant transformation matrix is given as T. Then we have the following relationship:

$$T(P_C) = Q_C \tag{1}$$

Note that due to the properties of reference objects, we have $P_T \subseteq P_C$ and $Q_T \subseteq Q_C$. Although our goal is to use the point cloud subset of the reference objects for the registration, some points from the areas enclosed by the reference objects are also preserved in the point cloud subset. Assuming the point cloud is well generated, the area enclosed by P_C is the same as that enclosed by Q_C. As the target objects from both point cloud datasets are the same, the location of P_T in P_C is the same as the location of Q_T in Q_C. Moreover, since T is linear, we have

$$T(P_T) = T(Q_T) \tag{2}$$

The transformation matrix generated by performing registration on the reference objects can properly align the target object. Therefore, instead of performing the registration on the entire point cloud, we can only perform registration on chosen stable objects to obtain the transformation matrix and then apply it to the entire point cloud.

As illustrated in Figure 1, the whole process was separated into two parts. The first part is to generate the 3D point clouds and 2D ortho-photos of different surveys based on the SfM software, such as Pix4D and Photoscan. The second part is our proposed workflow to register the point clouds of different surveys together. This part consists of three components.

The first is to extract the stable reference objects from the ortho-photo produced by the UAV-collected photos. We name it the reference-object-detection model.

The second component determines the bounding box coordinates of the reference objects in the point cloud. These coordinates are used to clip point clouds of the reference objects from the entire point cloud. We name it the coordinate converter model.

The last component is to perform the registration of the clipped point clouds of the reference objects and the resulting transformation matrix is then applied to the entire point cloud. We name it the segment-registration model.

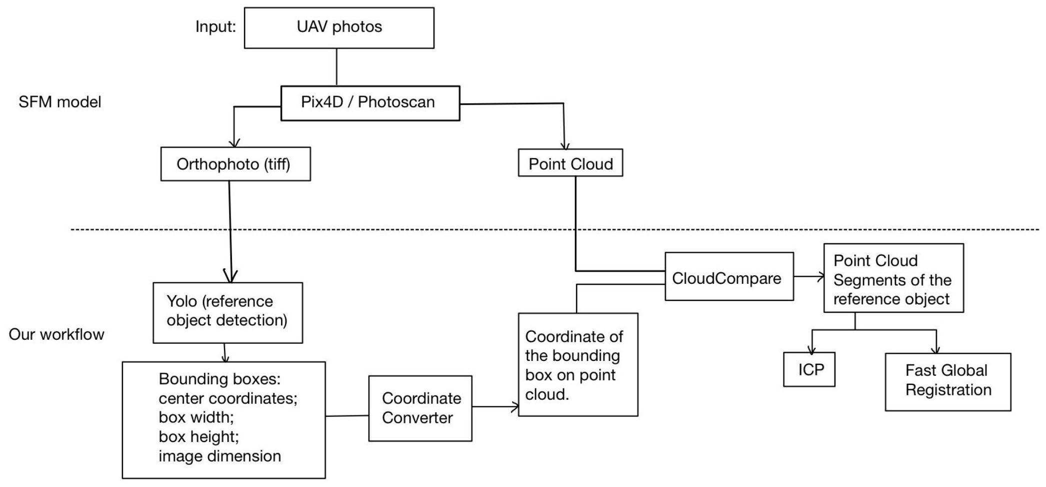

Input:
UAV photos
SFM model
Pix4D / Photoscan
Orthophoto (tiff)
Point Cloud
Our workflow
Yolo (reference object detection)
CloudCompare
Point Cloud Segments of the reference object
Coordinate of the bounding box on point cloud.
ICP
Fast Global Registration
Bounding boxes: center coordinates; box width; box height; image dimension
Coordinate Converter

Figure 1. Complete workflow of the proposed model.

3.1. Reference-Object-Detection Model

We use the ortho-photo generated via SfM software using UAV-collected photos as inputs for reference-object detection. The ortho-photo, usually in the tiff format, is a geometrically corrected representation of an area of interest with the same point cloud coordinate system. We use YOLO (you only look once) as the core algorithm of the detection model and Roboflow as the server to deploy the YOLO algorithm. Compared to the traditional object-detection algorithm, such as R-CNN, YOLO can employ a relatively simple CNN architecture to compute the detected object's bounding box directly. Generally, the YOLO algorithm performs faster regarding the object-detection area [26].

The reference-object-detection model first opens the ortho-photo using the Python Image library and runs a developed crop_center() function to keep the centre of the image. Then, the model keeps decompressing the image size until it fits the size that the Roboflow server can handle. After that, the image is sent to the AI object-detection server on Roboflow to extract the expectant object. The detection result is a JSON file consisting of two parts: the width and length of the compressed image and the detected object's bounding box width and length and its class name. After that, the model performs further processes on the JSON file, extracting the class names from the JSON file, setting them as the dictionary key, and displaying them on the user interface. Furthermore, under each key, it will store two lists, coordinates and dimensions. The coordinate list contains each object's central XY coordinates on the image, and the dimension list contains the width and length of each corresponding bounding box. We use these values combined with the values from the XYZ file and TFW file, which are both created during the generation of the point cloud, to calculate the three variables: box_bottom_left, photo_projected, and photo_bottom_left. Their explanations are given as follows:

- box_bottom_left contains the x,y-coordinate of the bottom left point in the point cloud dataset.
- photo_projected contains the length and width of the land shown in the cropped ortho-photo, measured in meters.
- photo_bottom_left contains the x,y-coordinate of the bottom left point of the land shown in the cropped ortho-photo.

Figure 2 illustrate the example dataset for object detection. Note that the coordinates calculated are based on the World Geodetic System. The XYZ file contains the x-, y-, and z-coordinates of all points in the point cloud dataset. By extracting the minimum value of them, we can find the box_bottom_left (denoted *bbl*) as follows:

$$bbl_x = \min(x) \qquad ; \qquad bbl_y = \min(y) \tag{3}$$

Apart from that, we also need to extract maximum and minimum values of the z-coordinate, which are needed in the crop stage later. The TFW file contains six parameters related to the properties of ortho-photo, namely the pixel size in the x-direction (denoted $pixel_x$), the pixel size in the y-direction (denoted $pixel_y$), the x-coordinate of the upper left corner (denoted pul_x), the y-coordinate of the upper left corner (denoted pul_y), the rotation parameter for the x-axis, and the rotation parameter concerning the y-axis. We only use the first four parameters. Note that the length and width of the entire ortho-photo is given by:

$$\text{length}_{\text{full}} = pixel_y \times \text{im\_length} \tag{4}$$

$$\text{width}_{\text{full}} = pixel_x \times \text{im\_width} \tag{5}$$

The value stored in photo_projected is calculated by:

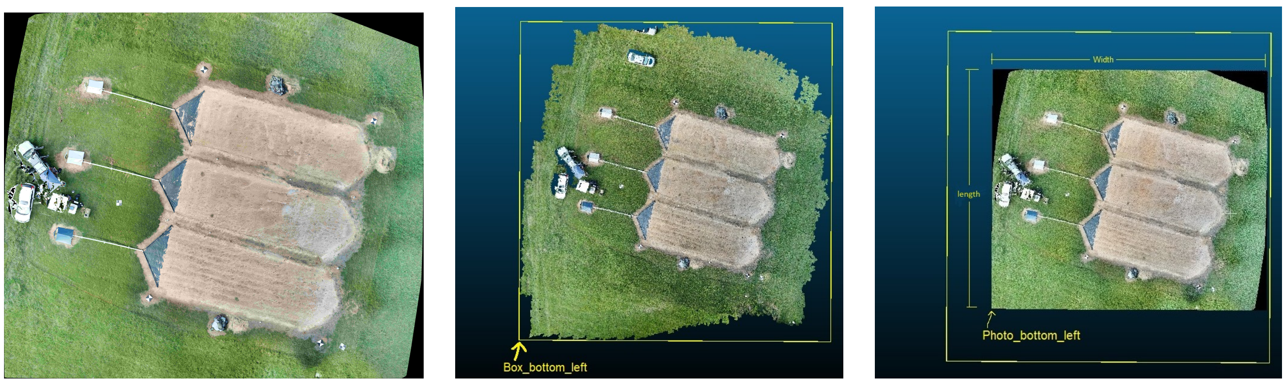
$$\text{length} = \text{length}_{\text{full}} \times \text{resize\_h} \tag{6}$$

$$\text{width} = \text{width}_{\text{full}} \times \text{resize\_w} \tag{7}$$

where im_length, im_width can be obtained from the properties of ortho-photo and resize_h, resize_w corresponds to the value used in the crop_center() function. The photo_bottom_left (denoted *pbl*) can be found via the following formulas:

$$pbl_x = pul_x + \frac{1 - \text{resize\_w}}{2} \times \text{width}_{\text{full}} \tag{8}$$

$$pbl_y = pul_y - \left(1 - \frac{1 - \text{resize\_h}}{2}\right) \times \text{length}_{\text{full}} \tag{9}$$



(**a**) Ortho-photo (**b**) Point Cloud (**c**) Ortho-photo on Point Cloud

Figure 2. The ortho-photo and point cloud of three erosion plots located in the University of Tennessee Plant Science unit of the East Tennessee AgResearch and Education Center. The ortho-photo (**a**) and point cloud (**b**) are generated by Pix4Dmapper. The point corresponding to box_bottom_left is marked on (**b**). (**c**) The position of the ortho-photo inside the point cloud. The value of height and width corresponding to photo_projected and the point corresponding to photo_bottom_left are marked in (**c**).

3.2. Coordinate Converter Model

From the previous component, we obtained the properties of bounding boxes for reference objects in the ortho-photo. These bounding boxes can be used to extract the point clouds of the detected reference objects. Before doing that, a noise removal algorithm is applied to remove outliers from the point cloud. The outliers or noise points are generated when employing the SfM software, such as Pix4D and Photoscan, for the reconstruction of the point cloud, due to the bad quality (overexposure, underexposure, or blurred images) of some photos (Figure 3). As previously mentioned, the 3D point cloud data of a geographic area are usually a large dataset. Employing traditional statistical approaches like distance-based or density-based calculations imposes a substantial computational burden [27] for large datasets. Furthermore, the density-based approach fails to remove a dense noise set, like the example shown in Figure 3, effectively. Hence, we develop a standard score method for noise removal. Note that this noise-removal method is restricted to flat land only; in this case, instead of calculating the Euclidean distances between each pair of points, we only consider the z-coordinates of the entire point cloud for noise removal. We make this noise-removal step an optional step. The users can also implement other methods for noise removal in other topographic settings.

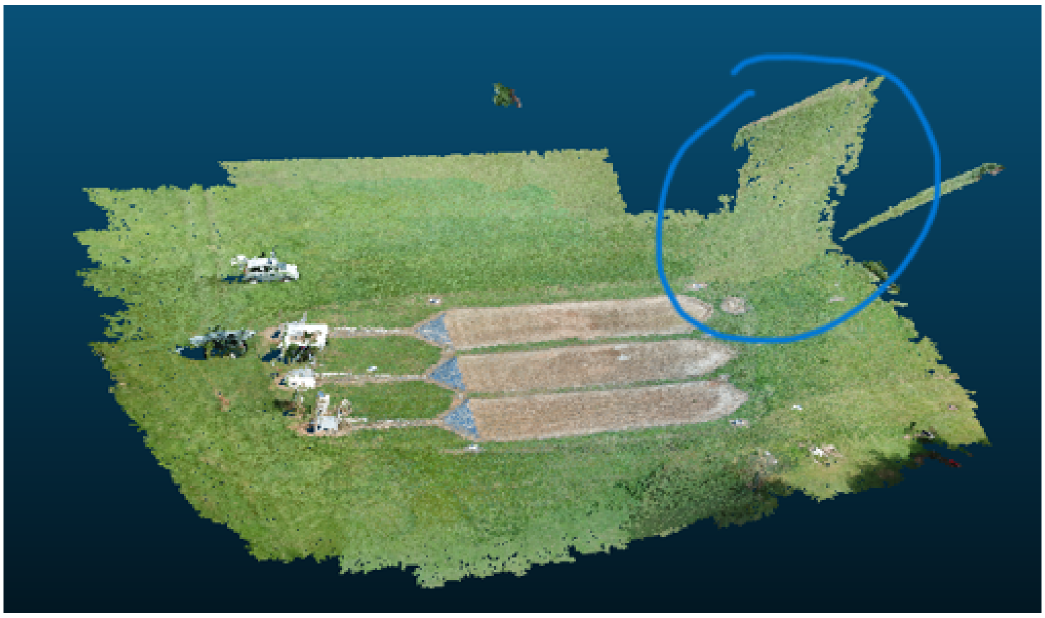


Figure 3. A point cloud of our test area generated by Pix4D with a dense noise set (the part circled in blue) because of bad photo quality.

Assume a point cloud A. Traversing all points p in A, we first calculate $\bar{z}$, the mean value of z-coordinates, using the formula:

$$\bar{z} = \sum_{p \in A} \frac{p_z}{N_A} \tag{10}$$

where p_z is the z-coordinate of point p, and N_A is the number of points in A. The standard deviation σ_z is given by:

$$\sigma_z = \sqrt{\frac{\sum_{p \in A} (p_z - \bar{z})^2}{N_A}} \tag{11}$$

and the cleaned point cloud A' is obtained by selecting the points that fall within a specified range of z-coordinate values. A formal definition is given as follows:

$$A' := \{p \in A \mid \; p_z \leq \bar{z} + 2 \times \sigma_z \text{ and } p_z \geq \bar{z} - 3 \times \sigma_z\} \tag{12}$$

Note that the multiplier of sigma can be modified if the target object for registration is also cropped out.

The width and length of the bounding boxes are given in pixels from the reference-object-detection model. It is necessary to convert these values to meters to be consistent with the coordinate system of the point cloud. Several calculations are developed for this conversion.

3.2.1. Projection

The coordinates of the detected objects in the reference-object-detection model only refer to the pixel coordinates (columns and rows) of the ortho-photo, not the geographic coordinates of the SfM reconstruction. Thus, we need a project function to map the pixel-based coordinates of the detected objects to the geographic coordinates (the UTM projection in this study) based on the World Geodetic System. The Coordinate Cropping Model will start when the user chooses the class names on the interface and consists of one primary function, crop_coordinate(). The crop_coordinate() is the function that controls the Coordinate Cropping Model's whole workflow. The crop_coordinate() function calculates the projected coordinates on the point cloud using the variable photo_bottom_left and box_bottom_left, and stores them in the array projected_reference_center and also the range of their corresponding bounding box is stored in the reference_box array. After that, we pass the projected_reference_box and reference_box arrays along with the Z range of the original point cloud to the function crop_las_file(). The point cloud cropping and generating process will be covered in the next part.

3.2.2. Cropping

This component employs an iterative process, analyzing tuples within the 'projected_reference_center' and 'reference_box' arrays. Each iteration entails reading the point cloud file via the 'pylas.read()' library, storing the point cloud data in the variable 'las'. Subsequently, a 2D NumPy array named 'points' is generated by vertically stacking the 'x', 'y', and 'z' coordinates of the points in the 'las' object. Each row in the 'points' array represents a 3D point. The array is then transposed using '.T' to ensure the appropriate shape. The model acquires half of the bounding box's x and y dimensions from the 'reference_box' list. With this information, the program computes the minimum and maximum bounding coordinates of a 3D box (bounding box) centered around specific coordinates within the point cloud, enabling filtering points within the bounding box. This is accomplished through the use of NumPy's logical comparison, where each point in the 'points' array is assessed to check if its coordinates (x, y, z) are greater than or equal to 'min_bounds' and less than or equal to 'max_bounds'. Subsequently, a Boolean mask ('mask') is applied to the 'las.points' attribute, which is a 2D array containing all point data, including x, y, z, and other attributes within the 'las' object. This process results in the retention of points exclusively within the specified bounding box, while discarding the remaining points (known as bounding box cropping).

3.3. Segment Registration Model

Two existing models are primarily applied in cloud registration: ICP and FGR. The two algorithms are conveniently integrated into CloudCompare. However, it is crucial to understand the principles behind these methods and effectively adjust the data and parameters to improve running time and registration accuracy.

We have developed comprehensive workflows for both FGR (Figure 4) and ICP (Figure 5) algorithms, aiming to enhance the clarity and comprehensibility of each step in the registration process. In terms of the data size, those two methods perform differently based on their principles.

The FGR method is renowned for its exceptional computational efficiency, especially when handling large point clouds. Unlike other registration algorithms with potentially higher time complexities, the FGR method maintains a linear time complexity in the order of points ($O(n)$). This advantage makes it particularly well suited for large-scale point cloud registration. Furthermore, the FGR method's iterative and data-parallel nature makes it highly amenable to parallelization, harnessing the power of multi-core processors or GPU acceleration. This parallel processing capability significantly accelerates the registration process, rendering the FGR method even more favorable for large-scale point cloud alignment tasks. As a result, the FGR method stands out as an ideal choice when dealing with extensive datasets, offering both computational efficiency and parallelization advantages.

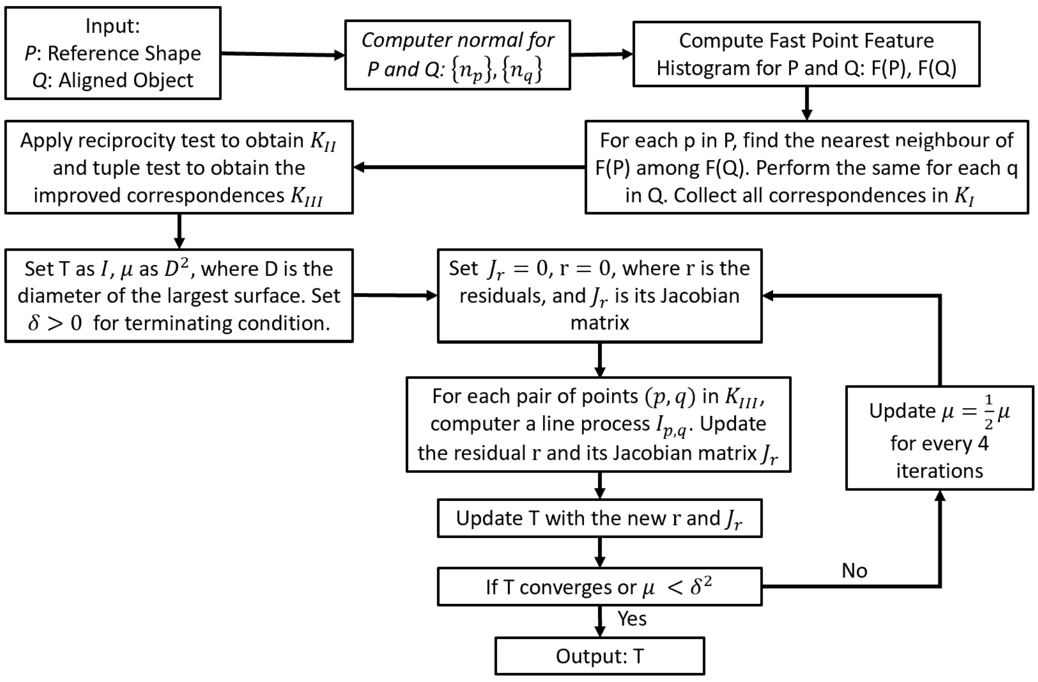

Input:
P: Reference Shape
Q: Aligned Object
Computer normal for P and Q: $\{n_p\}, \{n_q\}$
Compute Fast Point Feature Histogram for P and Q: F(P), F(Q)
For each p in P, find the nearest neighbour of F(P) among F(Q). Perform the same for each q in Q. Collect all correspondences in K_I
Apply reciprocity test to obtain K_{II} and tuple test to obtain the improved correspondences K_{III}
Set T as I, μ as D^2, where D is the diameter of the largest surface. Set $\delta > 0$ for terminating condition.
Set $J_r = 0$, r = 0, where r is the residuals, and J_r is its Jacobian matrix
For each pair of points (p, q) in K_{III}, computer a line process $l_{p,q}$. Update the residual r and its Jacobian matrix J_r
Update T with the new r and J_r
If T converges or $\mu < \delta^2$
Yes
No
Update $\mu = \frac{1}{2}\mu$ for every 4 iterations
Output: T

Figure 4. Workflow of FGR.

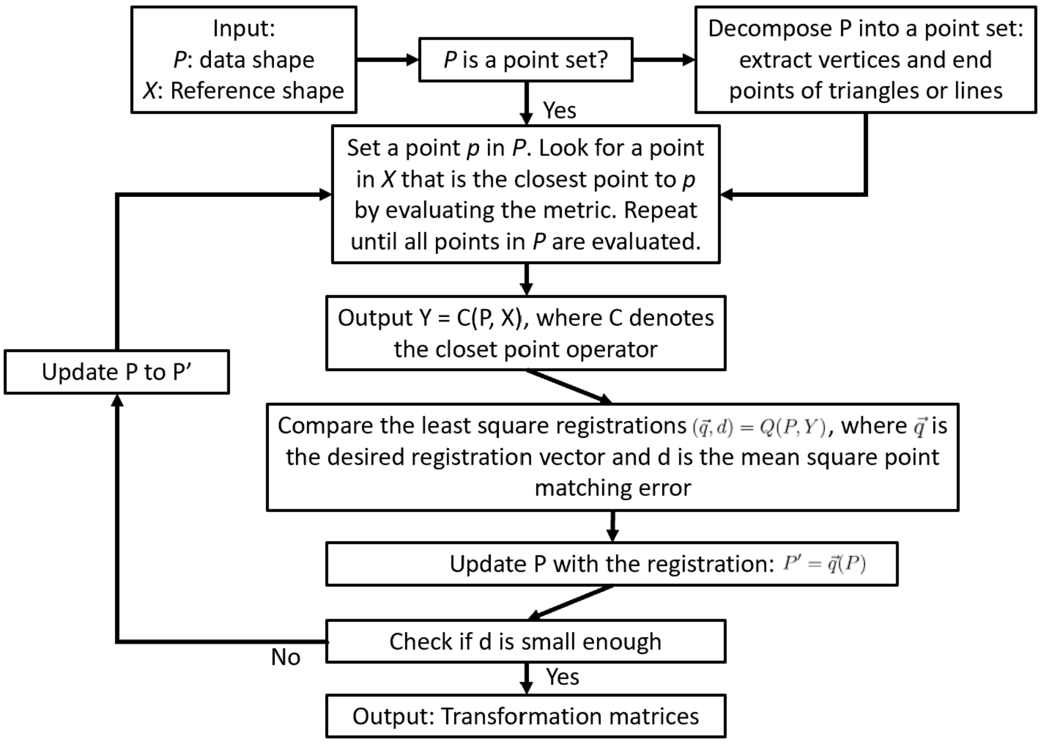

Input:
P: data shape
X: Reference shape
P is a point set?
Yes
Decompose P into a point set: extract vertices and end points of triangles or lines
Set a point p in P. Look for a point in X that is the closest point to p by evaluating the metric. Repeat until all points in P are evaluated.
Output Y = C(P, X), where C denotes the closet point operator
Compare the least square registrations $(\vec{q}, d) = Q(P, Y)$, where $\vec{q}$ is the desired registration vector and d is the mean square point matching error
Update P with the registration: $P' = \vec{q}(P)$
Check if d is small enough
No
Yes
Update P to P'
Output: Transformation matrices

Figure 5. Workflow of Iterative Closest Points.

ICP, in contrast, performs better when there are fewer points to match and optimize. ICP converges more quickly to an acceptable alignment. In each iteration, the algorithm aims to minimize the distance between corresponding points, and with fewer points, this process becomes more efficient, leading to faster convergence. The time complexity of ICP is generally linear concerning the number of points in the datasets. Therefore, when fewer points exist, the algorithm's computational load is reduced, making it more efficient for small point clouds. In addition, finding correspondences between two point clouds is a crucial step in the ICP algorithm. With fewer points, the search for corresponding points becomes less complex and time-consuming, enhancing the accuracy of the alignment. The quality of the initial alignment estimate can influence the outcome of ICP as well.

Based on the above discussion, the segment registration model serves the function of determining the appropriate registration method to apply by evaluating the attributes of the cropped point cloud. As described in Section 4, we recommend that when the cropped point cloud contains a relatively small number of points, ICP is a better choice for registration. Conversely, if the cropped point cloud is substantial in size, FGR is a better option. Although the time complexity of the ICP and the FGR methods remains $O(n^2)$ and $O(n\log n)$, respectively, the sample size n is greatly reduced by just using the point clouds of the reference objects. Hence, the operation time can be improved.

4. Experimental Results

We evaluate the performance of our proposed workflow based on the UAV data collected from three erosion plots at the Plant Science unit of the East Tennessee Research and Education Center (ETREC), University of Tennessee. With an approximate length of 21 m, a width of 6 m, and a slope of 15 percent, these plots were designed to be hydrologically isolated with a sediment capturing system installed at the bottom [28,29]. These plots were used for testing the detection limit of TLS and UAVs on bare hillslope erosion [28]. They were maintained to be largely free of vegetation [29]. Several sets of UAV data were collected from these plots from 2019 to 2021. Six concrete mounting points were installed at the corners of each plot as fixed locations for placing ground targets. Four black/white ground targets were placed on the four corners of these plots during each UAV flight. A DJI Mavic Pro drone was used to map the area with a double grid flight mission designed using the Pix4DCapture App. This flight mission was designed to enhance the 3D reconstruction of the topography. In each survey, the drone flew at a height of approximately 10 m above the ground, achieving a ground sampling distance (GSD) of the photo of about 0.35 cm/pixel. After collecting the photos, we used the Pix4DMapper to reconstructed the 3D point clouds and ortho-photos, based on the default settings for standard 3D mapping. Specifically, Pix4DMapper used a photo-size-reduction factor of $\frac{1}{2}$ to speed up the reconstruction. This setting also allowed the average spacing of the point cloud to be tailored according to the resolution of the UAV photos, thereby facilitating a more accurate and detailed 3D reconstruction. The average spacing of the point cloud generated using this setting is about 1.2 to 1.4 cm.

In our test cases, the reference objects are chosen to be House (tent), Pipe, and Drone targets. The values of resize_h and resize_w corresponding to the crop_center() function are both set to 70%. To cope with the condition of the survey site, several steps are added beyond the Coordinate Converter Model to handle different detected objects. For instance, when the object identified is a 'House', the model processes the object as mentioned in the previous section. Upon the completion of the 'House' object's processing, the program proceeds to handle the other two types of objects, namely 'Drone targets' and 'Pipe'. Specifically, white color segmentation and noise removal are executed for the 'Drone targets' and 'Pipe' objects. This is particularly significant for the 'Pipe' point cloud, as it is susceptible to noisy points introduced during bounding box cropping.

Further processing is also performed if the object is a 'Drone targets'. Specifically, the model calculates the center point of the segmented point cloud (white points). Then, the center point is returned to the unsegmented drone point cloud. Leveraging the NumPy 'argpartition()' function, the indices of the 500 points closest to the center are identified. These indices are then used to crop the sub-point cloud from the drone point cloud and store it in the predefined directory. After that, all the cropped drone point clouds are combined into one for future transformation matrix calculation convenience.

We test the performance of our proposed workflow based on four sets of data, which record the topographic change of the erosion plots over about one year from 4 September 2020 to 8 September 2021. In each set of data, we use approximately 300 pictures taken by the same DJI Mavic Pro to generate the point cloud. The details of the point cloud are listed in Table 1. The erosion plots experienced minor topographic changes over this period due to the rainfall events. During this period, the sediments captured at the bottoms of these three erosion plots ranged from 540 to 1300 kg, corresponding to the average erosion depths (topographic changes) of 2.3 to 5.1 mm over the plots. Note that two UAV surveys (set 1 and set 2) were conducted on the same date of 4 September 2020, allowing for the assessment of the registration accuracy of the two point clouds from the 'same' topography. With the whole point cloud, we can detect and extract houses, pipes, and drone targets. Figure 6 shows how these objects are marked on the ortho-photo. In all four datasets, there are four black/white targets, three pipes and three houses (tents) to choose from as reference objects. In Figure 6 , the purple boxes are the bounding boxes for the Houses (Tents), and the red bounding boxes with blue labels represent the Pipes. Last, the green bounding boxes with green labels are for the drones located at the four corners of the plots. Each bounding box contains a label on the left upper corner, which contains the class name of its objects, which is House, Pipe, and Drone for our case. Besides the class name, it also shows the confidence level of the detected objects. An enlarged version of the same figure is shown in Appendix A. The object-detection process takes 6 s, and the extraction of the partial point cloud of these reference objects takes around 10 s for the test data.

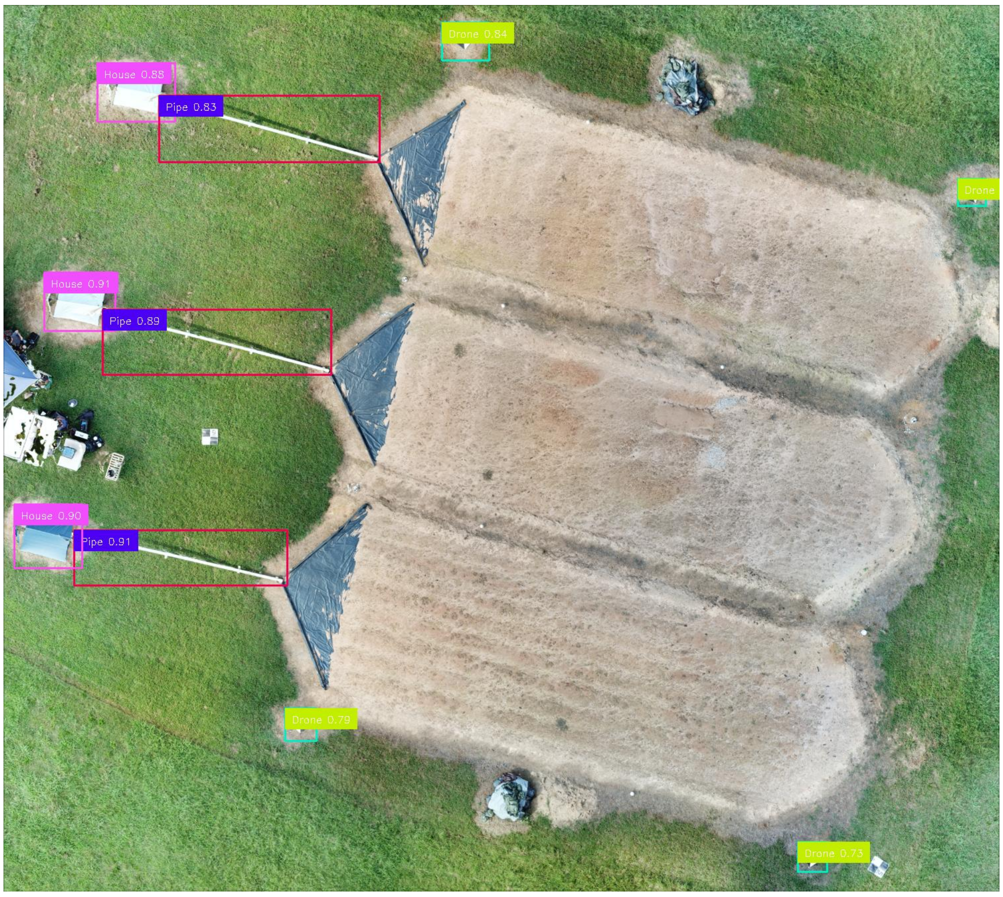
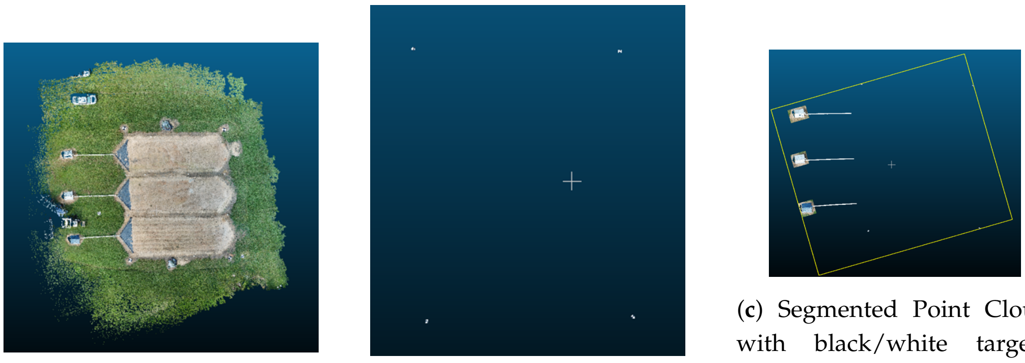


Figure 6. Reference objects detected from the ortho-photo. The object marked as the drone class are the black/white ground targets.

Table 1. The four datasets used for the performance test.

	Number of Points	Average Point Spacing (m)
4 September 2020 (set 1)	16,355,234	0.011
4 September 2020 (set 2)	17,838,314	0.012
23 April 2021	16,536,909	0.014
8 September 2021	15,201,529	0.012

Both the ICP and FGR methods are performed on the partial point cloud of extracted reference objects. Since ICP is a local method while FGR is a global method, we use the segmented point cloud containing only black/white targets for ICP, while we use the segmented point cloud formed with black/white targets, houses, and pipes for FGR. Figure 7 shows the segmented point cloud that we extracted for registration.



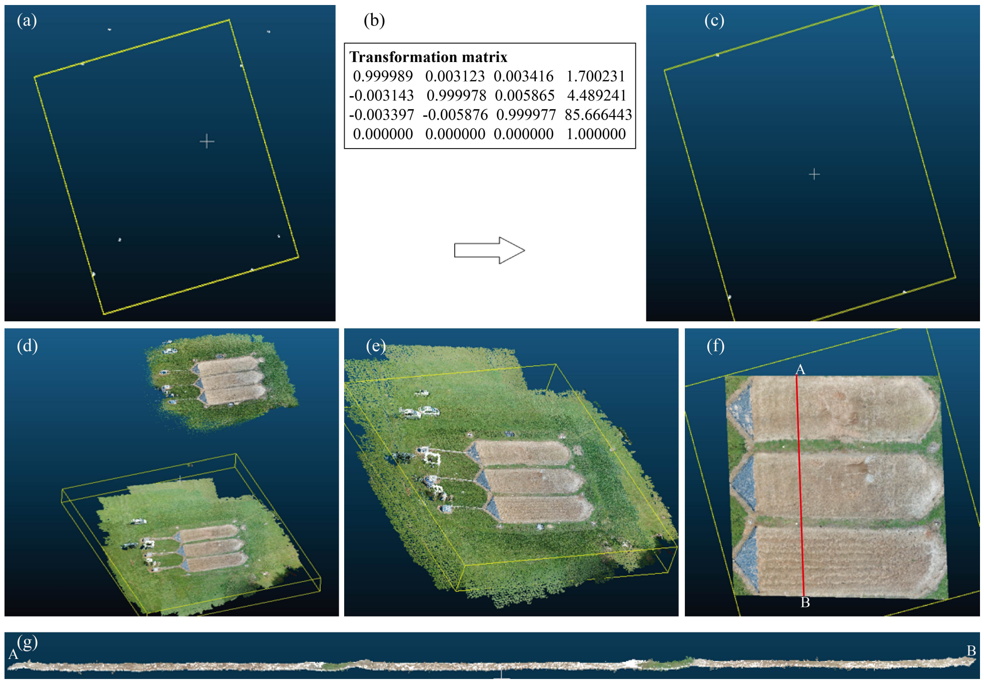
(**a**) Original Point Cloud

(**b**) Segmented Point Cloud with black/white targets

(**c**) Segmented Point Cloud with black/white targets, houses, and pipes

Figure 7. The original point cloud (**a**) and the segmented point clouds used for ICP (**b**) and FGR (**c**) registrations. The "+" indicates the center location of the view in CloudCompare.

Since our focus of the registration is on the three erosion plots and the cars that we parked were at different locations on different dates of the survey, which affect the accuracy assessment of the registration, we crop the point clouds to the extent of the three erosion plots as illustrated in Figure 8f for the accuracy assessment. We name it the entire cropped point cloud in the following part of the paper. In general, running ICP on the partial point cloud takes around 5 s, while performing ICP on the entire cropped point cloud takes around 50 s. For the FGR method, we have to compute the normals of the partial point cloud before registration. We compared the outcome of normal approximation by employing the Hough Transform [30] and the built-in calculator in CloudCompare and the latter one performs better. The normal approximation process takes around 40 s. Once the normals are derived, the FGR method is applied, which takes around 75 s to accomplish. However, it is essential to note that during a preliminary test on the entire cropped point cloud, the normal approximation process alone consumes about 40 min and the registration stage takes more than five hours to operate. Hence, we do not consider the FGR on the entire cropped point cloud as a practical approach for real-time applications and exclude it from our tests. The test results are listed in Tables 2–4. We denote D as the point cloud formed by drone targets, P as the point cloud formed by pipes, H as the point cloud formed by houses, and E as the entire cropped point cloud. Note that we need to merge all the desired reference objects into one point cloud before performing registration. For each pair of point clouds, we evaluate the mean distance, calculated using Euclidean distance, between each pair of erosion plots and make the maximum value the **Distance**. Moreover, the **Standard Deviation** (SD) is calculated using the pairs with the maximum mean distance. All distance and standard deviation values listed in the tables are based on the calculations for the entire cropped point cloud.



```
Transformation matrix
 0.999989  0.003123 0.003416  1.700231
-0.003143  0.999978 0.005865  4.489241
-0.003397 -0.005876 0.999977 85.666443
 0.000000  0.000000 0.000000  1.000000
```

Figure 8. The two point clouds (4 September 2020 (set 2) and 8 September 2021) at each stage of the ICP registration based on our proposed workflow: The point cloud of segmented objects (drone ground targets, pipes, and houses) before registration (**a**). A screenshot of the transformation matrix for the registration in CloudCompare (**b**). The registered point clouds of the drone ground targets (**c**). (**d**,**e**) The positions of all the point clouds before and after applying the transformation matrix, respectively. (**f**) The three erosion plots after registration and (**g**) a swath profile along the red line marked on (**f**). A and B are the start and end points of the swath profile. The "+" indicates the center location of the view in CloudCompare.

Table 2. The data collected on 8 September 2021 are aligned with those collected on 23 April 2021. D denotes the extracted point cloud of drone targets; D, P, and H denote the extracted point cloud of pipes, drone targets, and houses; E denotes the entire (original) point cloud.

	ICP on D	ICP on E	FGR on D, P and H
Distance (m)	0.0018	0.6286	0.0419
Standard Deviation (m)	0.0155	0.7997	0.0639

Table 3. The data collected on 4 September 2020 (set 2) are aligned with those collected on 8 September 2021. D, P, H, E are the same as Table 2.

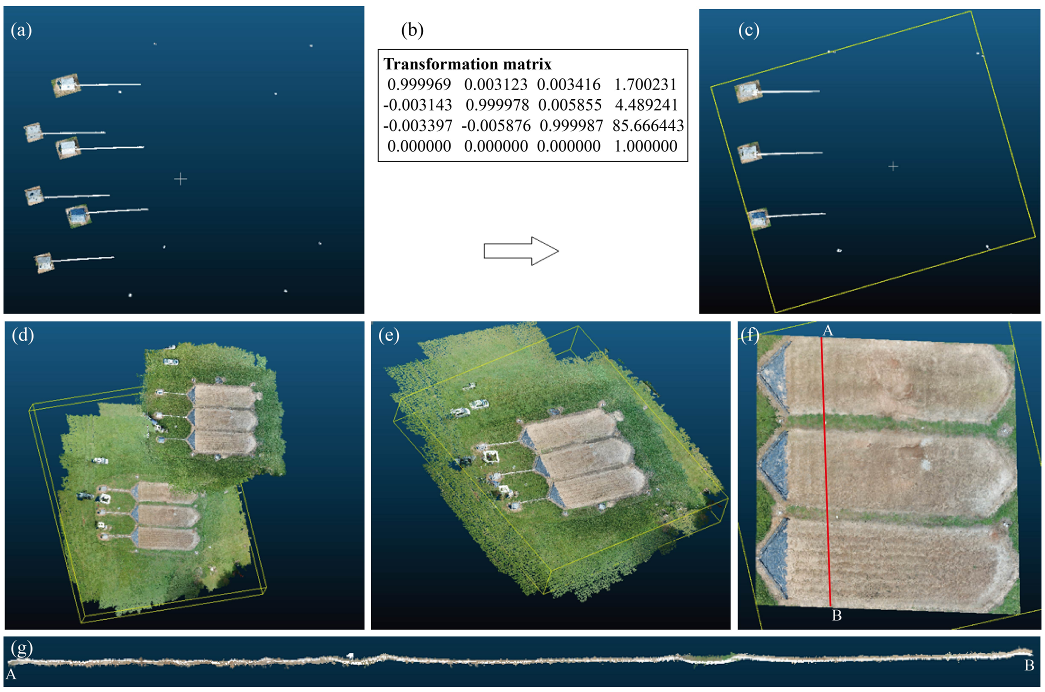
	ICP on D	ICP on E	FGR on D, P and H
Distance (m)	0.0015	0.9173	0.0396
Standard Deviation (m)	0.0144	1.0871	0.0623

Table 4. Two sets of data are both collected on 4 September 2020; 4 September 2020 (set 1) is aligned with 4 September 2020 (set 2). D, P, H, E are the same as Table 2.

	ICP on D	ICP on E	FGR on D, P and H
Distance (m)	0.0001	0.0005	0.0329
Standard Deviation (m)	0.0041	0.0086	0.0595

Figure 8 showcases the results (the alignment and the cross-section of the plots) obtained from our proposed workflow using the ICP registration. Note that the two point clouds are apparently in very different locations in CloudCompare with a large offset (about 78.7 m for this case). This offset does not indicate the real distance between the two point clouds. CloudCompare applies a global shift to convert a point cloud with big coordinates (>100,000) to a local coordinate system with smaller coordinates to keep the original precision of the coordinates and speed up the computation. The application of this global shift may cause the two point clouds from a same area to be apart from each other due to the impact of noise points on the extent of each point cloud. Please check the CloudCompare online manual for details. These results indicate an effective registration of the two point clouds based on the ICP registration using the black/white targets. The mean point–point cloud distances are about 0.15 cm (SD = 1.4 cm) to 0.17 cm (SD = 1.5 cm) for the point clouds of different times. The mean point–point cloud distance is only 0.01 cm (SD = 0.4 cm) for the two point clouds on September 4, 2020, suggesting a highly accurate registration.

Figure 9 demonstrate the registration results using the FGR method. The registration also looks effective although the mean point–point distances are about 3 to 4 cm with standard deviations of around 6 cm (Tables 2–4).



Transformation matrix

0.999969	0.003123	0.003416	1.700231
-0.003143	0.999978	0.005855	4.489241
-0.003397	-0.005876	0.999987	85.666443
0.000000	0.000000	0.000000	1.000000

Figure 9. The two point cloud files (4 September 2020 (set 2) and 8 September 2021) at each stage of the FGR based on our proposed workflow. The point cloud of segmented objects (drone ground targets, pipes, and houses) before registration (**a**). A screenshot of the transformation matrix for the registration in CloudCompare (**b**). The registered point clouds of segmented objects (**c**). (**d**,**e**) The positions of all the point clouds before and after applying the obtained transformation matrix, respectively. (**f**) The three erosion plots after registration and (**g**) a swath profile along the red line marked on (**f**). A and B are the start and end points of the swath profile. The ”+” indicates the center location of the view in CloudCompare.

We apply the ICP method directly to the entire cropped point cloud (Figure 10a). As illustrated, the registration is not effective with a noticeable shift observed for the grey triangular object (collection triangle), demonstrating that directly applying the ICP method to the entire point cloud fails to produce satisfactory registration. The ICP registration using the segmented point cloud, consisting of black/white targets, pipes, and houses, also falls short of our expectations with a mean distance of 0.425 m between the three erosion plots of the two point clouds (Figure 10b).

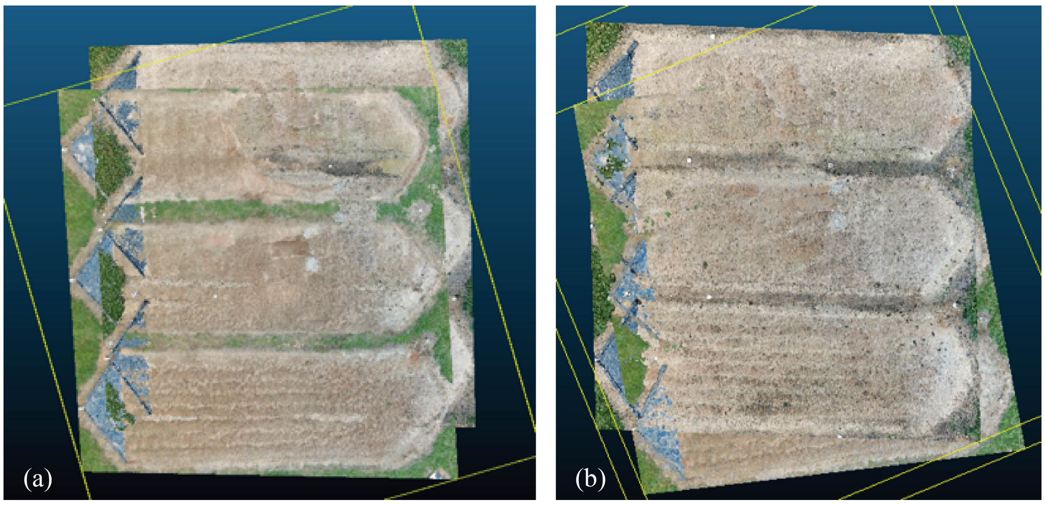


Figure 10. The cropped three erosion plots after performing the ICP registration on all the point clouds (**a**) and on the point clouds of segmented drone ground targets, pipes, and houses (**b**). Both registrations are carried out by aligning the data collected on 4 September 2020 (set 2) with those collected on 8 September 2021. For (**a**), the distance between the 2 sets is about 0.917 m (Table 3). For (**b**), the distance between the 2 sets is about 0.425 m.

5. Discussion

The average spacing of the point clouds from our test area is about 1.2 to 1.4 cm, indicating that any detailed horizontal changes less than this average spacing are likely not to be detected. However, it is hard to evaluate the accuracy of the changes in elevation. As listed in Table 4, the cloud-to-cloud distance from the two point clouds generated using the two UAV flights on the same day reached 0.1 mm for the registration based on the four drone targets (ICP-D) and 0.5 mm for the registration based on the entire cropped point cloud (ICP-E). Since the registration is performed on two datasets on the same day, we can treat the reconstructed topography as the same surface and neglect the impact of soil erosion and grass changes. Thus, the cloud-to-cloud distance of 0.1 mm for ICP-D and 0.5 mm for ICP-E only represent the registration error, indicating a very high registration accuracy in horizontal distance and elevation. The cloud-to-cloud distance from the point clouds generated using the two UAV flights about one year apart includes both registration error and the impacts of soil erosion on the surface and different grass conditions outside of the plots. In the case listed in Table 3, the cloud-to-cloud distance reached 1.5 mm for ICP-D, about 15 times higher than the registration error of 0.1 mm from the same day. It is still a very accurate registration because even though we could put the black/white ground targets on the same spots of the concrete mounting points, the orientation and slope of each target were slightly different each time, affecting the registration. In addition, the grass conditions near the plots were also slightly different during the two survey periods and the soil surface within the plots also experienced 2.3 to 5.1 mm erosion on average based on the collected sediments from each plot. All these factors affect the calculated cloud-to-cloud distance. Note that the cloud-to-cloud distance of 1.5 mm is at the same level of magnitude of the soil surface changes of the three plots (2.3 to 5.1 mm), suggesting that the UAV-collected data may be capable of detecting mm-level detailed surface changes. We acknowledge that the purpose of this study is the registration of point clouds, and the quantification of detailed soil surface changes is beyond the scope of this study. We recommend more work in the future to quantify detailed soil surface changes after registering the point clouds to a high level of accuracy using our proposed workflow.

Among the test results, the application of the ICP method on the partial point cloud of black/white targets stands out as the most effective. It outperforms the others in both time efficiency and registration accuracy, with an average distance smaller than the average point spacing of the datasets (Table 1). These registration targets were placed at the same concrete mounting points during each survey, which is relatively stable compared to the tents covering the sample collection systems (houses). The pipes have relatively stable features, but extracted pipe point clouds include a significant amount of points from the nearby grasses. Additional cleanup of the pipe point clouds may improve the registration accuracy. In addition, the use of only black/white targets reduced the size of the point clouds for registration, reducing the chances of encountering a local minima problem. In contrast, the error incurred when applying the ICP to the entire point cloud is about 600 times greater than the registration applied solely to black/white targets. This indicates that applying the ICP method to a large point cloud dataset is not stable.

Regarding the FGR method, it performs significantly faster under our proposed workflow, with a reasonable error. This advantage becomes particularly important when dealing with large datasets, where processing time plays a critical role in real-world applications. As shown in Tables 2 and 3, applying the FGR method with our proposed workflow results in a lower error when compared with the registrations using the ICP method on the entire point cloud. It is possible that applying the FGR method to the entire point cloud may yield a slightly higher accuracy. However, it may take a significantly long time to process the registration for the entire dataset. Thus, we believe that it is more practical to apply the FGR method under our workflow.

In our test cases, the black/white targets prove to be an optimal choice for reference objects, enabling the successful application of the ICP method within our workflow. This is due to their fixed positional nature and suitability for segmentation, reducing the negative impact of noises and non-target points on the registration. However, it is important to note that this approach is contingent on the availability of such objects. In scenarios where such objects are absent, the FGR method may be more reliable in the registration.

Our test data were collected from a relatively flat hillslope. However, we believe that this AI-based workflow can apply to other topographic settings like steep topography, as long as stable ground target objects are detectable. The transformation matrix is generated from the registration of partial point clouds of detected stable objects, which remain unaffected by steep topography. We employed a simple Z-threshold approach to remove the noise because of its efficiency in our test data. This noise-removal method may not be suitable for cases involving steep topography, as the variation of z-values becomes much larger. This noise-removal step is optional and the users can implement other methods for noise removal in those topographic settings.

Despite the good performance of our model on large dataset registration, improvements are still needed to enhance its capabilities. First, the object-detection model can be trained further so that it can detect a wider variety of objects that may exist in different topographic settings. Presently, the model is only trained to detect three types of objects, namely the black/white target for UAV drones, pipes, and houses (tents). However, these targets may not appear in other areas for registration. Furthermore, we can employ parallel computing techniques to optimize the efficiency of our model. With parallel processing power, we can magnificently speed-up tasks such as noise removal while mitigating the risk of buffer overflow at the same time. Finally, we can make this workflow more accessible and user-friendly by building a plugin in CloudCompare with more customized interfaces. At present, users are required to manually apply the resultant transformation matrix to register the entire point cloud, which can be time-consuming. Automating all steps related to this AI-based workflow could streamline the registration process and significantly improve the usability and efficiency of this workflow.

6. Conclusions

The Iterative Closest Point (ICP) and Fast Global Registration (FGR) algorithms are common approaches used for point cloud registration. However, they are not specifically designed to handle large point cloud datasets. To address this limitation, we develop an AI-based workflow to extract stable reference objects and then only use the partial point clouds from the stable reference objects to enable accurate and effective registration of large point cloud data. This workflow ensures a more controlled and focused registration process, specifically tailored to the reference objects of interest. This targeted approach minimizes the possibility of errors or inaccuracies, contributing to a more reliable and robust registration.

One of the key advantages of our proposed workflow is its flexibility. Users are allowed to select their own reference objects for the registration process. By incorporating both ICP and FGR methods into our proposed workflow and applying them to smaller point cloud segments, we can efficiently align point clouds while mitigating the risk of encountering local minima issues. Although this workflow is demonstrated for the registration of the same erosion plots with a relatively flat topography at different times, it has the potential to be used for the registration of UAV-produced point clouds in other topographic settings.

Author Contributions: Conceptualization, Y.L.; methodology, Y.F. and K.L.L.; software, Y.F. and K.L.L.; validation, Y.F. and K.L.L.; formal analysis, Y.F. and K.L.L.; investigation, Y.F. and K.L.L.; resources, Y.L. and K.L.W.; data curation, Y.L.; writing—original draft preparation, Y.F., K.L.L. and Y.L.; writing—review and editing, Y.F., K.L.L. and Y.L.; visualization, Y.F. and K.L.L.; supervision, Y.L. and K.L.W.; project administration, Y.L. and K.L.W.; funding acquisition, K.L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Science Foundation (NSF) under the Research Experiences for Undergraduates (REU) with the award number 2020534, with additional support from the National Institute of Computational Sciences at University of Tenneessee, Knoxville.

Data Availability Statement: The data related to this study are available on request from the authors.

Acknowledgments: The authors thank Yuzhou Chen for his collaboration in the improvement of the model. The authors also like to thank Nathan McKinney, Gene Bailey, Daniel Yorder, and Wesley Wright for setting up the erosion plots and collecting the UAV data.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Detected Image

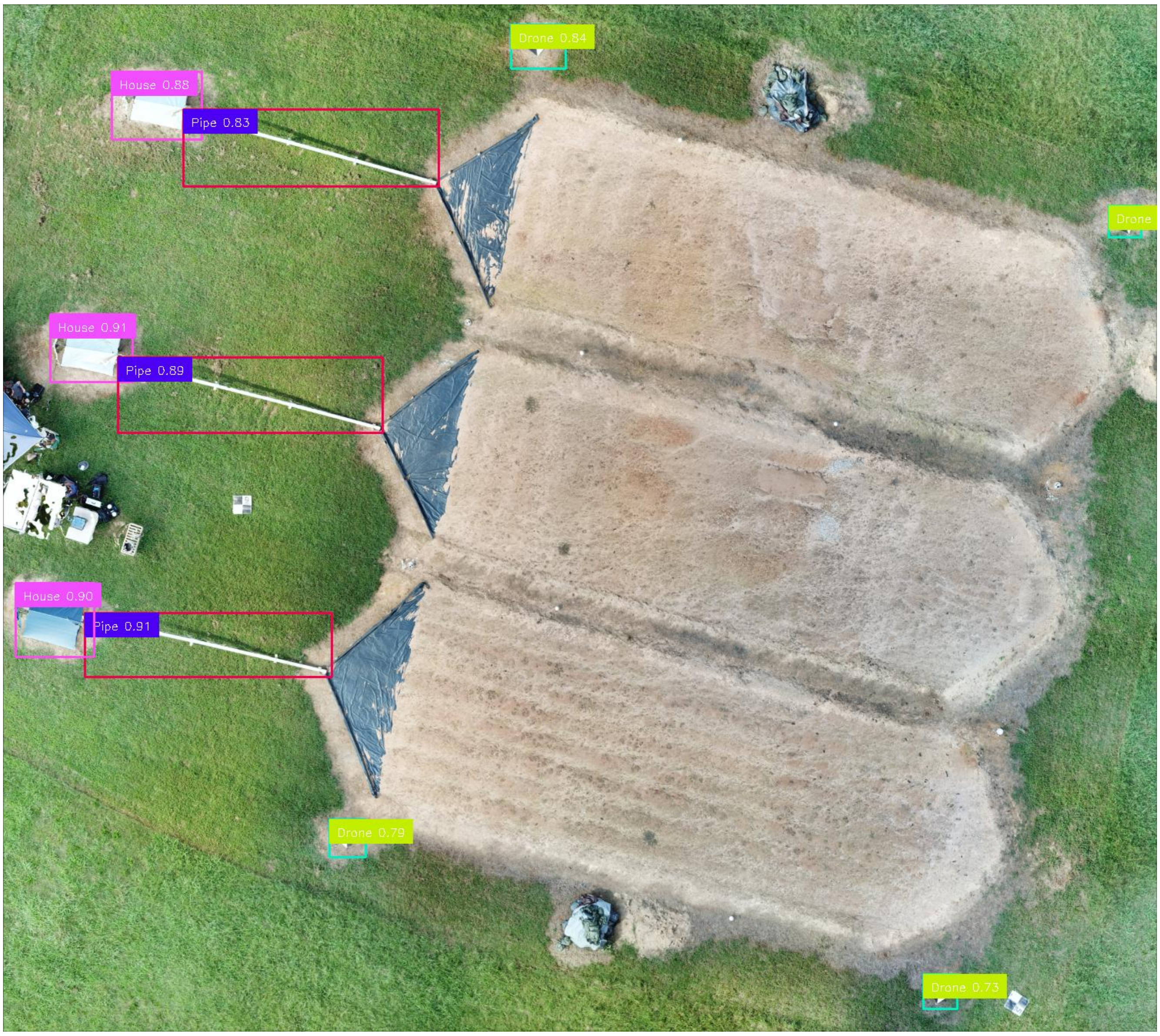


Figure A1. This is an enlarged version of the detected_image demonstrating the result of object detection.

References

1. Eltner, A.; Schneider, D. Analysis of different methods for 3D reconstruction of natural surfaces from parallel-axes UAV images. *Photogramm. Rec.* **2015**, *30*, 279–299. [CrossRef]
2. Cândido, B.M.; Quinton, J.N.; James, M.R.; Silva, M.L.N.; de Carvalho, T.S.; de Lima, W.; Beniaich, A.; Eltner, A. High-Resolution Monitoring of Diffuse (Sheet or Interrill) Erosion Using Structure-from-Motion. *Geoderma* **2020**, *375*, 114477. [CrossRef]
3. d'Oleire-Oltmanns, S.; Marzolff, I.; Peter, K.D.; Ries, J.B. Unmanned aerial vehicle (UAV) for monitoring soil erosion in Morocco. *Remote Sens.* **2012**, *4*, 3390–3416. [CrossRef]
4. Pavlis T.L.; Serpa L.F. Accuracy of Structure-from-Motion/Multiview Stereo Terrain Models: A Practical Assessment for Applications in Field Geology. *Geosciences* **2023**, *13*, 217. [CrossRef]
5. Candiago, S.; Remondino, F.; De Giglio, M.; Dubbini, M.; Gattelli, M. Evaluating multispectral images and vegetation indices for precision farming applications from UAV images. *Remote Sens.* **2015**, *7*, 4026–4047. [CrossRef]
6. Patil, M.; Saha, A.; Pıngale, S.M.; Rathore, D.S.; Goyal, V.C. Identification of potential zones on the estimation of direct runoff and soil erosion for an ungauged watershed based on remote sensing and GIS techniques. *Int. J. Eng. Geosci.* **2023**, *8*, 224–238. [CrossRef]
7. Li, M.; Nan, L.; Smith, N.; Wonka, P. Reconstructing building mass models from UAV images. *Comput. Graph.* **2016**, *54*. [CrossRef]

8. Yakar, M.; Dogan, Y. 3D Reconstruction of Residential Areas with SfM Photogrammetry. In Proceedings of the 1st Springer Conference of the Arabian Journal of Geosciences (CAJG-1), Sousse , Tunisia, 12–15 November 2019; pp. 73–75.
9. Fonstad, M.A.; Dietrich, J.T.; Courville, B.C.; Jensen, J.L.; Carbonneau, P.E. Topographic structure from motion: A new development in photogrammetric measurement. *Earth Surf. Process Landforms* **2012**, *38*, 421–430. [CrossRef]
10. Smith, M.W.; Carrivick, J.L.; Quincey, D.J. Structure from motion photogrammetry in physical geography. *Prog. Phys. Geogr. Earth Environ.* **2015**, *40*, 247–275. [CrossRef]
11. Eltner, A.; Kaiser, A.; Castillo, C.; Rock, G.; Neugirg, F.; Abellán, A. Image-based surface reconstruction in geomorphometry—merits, limits and developments. *Earth Surf. Dyn.* **2016**, *4*, 359–389. [CrossRef]
12. Javernick, L.; Brasington, J.; Caruso, B. Modeling the topography of shallow braided rivers using Structure-from-Motion photogrammetry. *Geomorphology* **2014**, *213*, 166–182. [CrossRef]
13. Wójcik, A.; Klapa, P.; Mitka, B.; Piech, I. The use of TLS and UAV methods for measurement of the repose angle of granular materials in terrain conditions. *Measurement* **2019**, *146*, 780–791. [CrossRef]
14. Ouédraogo, M.M.; Degré, A.; Debouche, C.; Lisein, J. The evaluation of unmanned aerial system-based photogrammetry and terrestrial laser scanning to generate DEMs of agricultural watersheds. *Geomorphology* **2014**, *214*, 339–355. [CrossRef]
15. Candan, L.; Kaçar, E. Methodology of real-time 3D point cloud mapping with UAV lidar. *Int. J. Eng. Geosci.* **2023**, *8*, 301–309. [CrossRef]
16. Karatas, L.; Dal, M. Deterioration analysis of historical village house structure in Mersin Kanlıdivane archaeological area by UAV method. *Mersin Photogramm. J.* **2023**, *5*, 32–41. [CrossRef]
17. Barbasiewicz, A.; Widerski, T.; Daliga, K. The analysis of the accuracy of spatial models using photogrammetric software: Agisoft Photoscan and Pix4D. In *E3S Web of Conferences* ; EDP Sciences: Les Ulis, France, 2018; Volume 26, p. 00012. [CrossRef]
18. Yilmaz, C.S.; Yilmaz, V.; Güngör, O. Investigating the performances of commercial and non-commercial software for ground filtering of UAV-based point clouds. *Int. J. Remote Sens.* **2018**, *39*, 5016–5042. [CrossRef]
19. Alidoost, F.; Arefi, H. Comparison of UAS-based photogrammetry software for 3D point cloud generation: A survey over a historical site. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *IV-4/W4*, 55–61. [CrossRef]
20. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
21. Yang, J.; Li, H.; Campbell, D.; Jia, Y. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 2241–2254. [CrossRef]
22. Zhou, Q.; Park, J.; Koltun, V. Fast global registration. In Proceedings of the 14th European Conference (ECCV 2016), Amsterdam, The Netherlands, 11–14 October 2016; pp. 766–782.
23. Jost, T.; Hügli, H. Fast ICP algorithms for shape registration. In Proceedings of the 24th DAGM Symposium, Zurich, Switzerland, 16–18 September 2002; pp. 91–99.
24. Fitzgibbon, A.W. Robust registration of 2D and 3D point sets. *Image Vis. Comput.* **2002**, *21*, 1145–1153. . [CrossRef]
25. Granger, S.; Pennec, X. Multi-scale EM-ICP: A fast and robust approach for surface registration. In Proceedings of the 7th European Conference on Computer Vision (ECCV 2002), Copenhagen, Denmark, 28–31 May 2002; pp. 418–432.
26. Joiya, F. Object detection: Yolo vs Faster R-CNN. *Int. Res. J. Mod. Eng. Technol. Sci.* **2022**, *4*, 1911–1915. [CrossRef]
27. Hodge, V.J.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
28. Bailey, G.; Li, Y.; McKinney, N.; Yoder, D.; Wright, W.; Washington-Allen, R. Las2DoD: Change Detection Based on Digital Elevation Models Derived from Dense Point Clouds with Spatially Varied Uncertainty. *Remote Sens.* **2022**, *14*, 1537. [CrossRef]
29. Bailey, G.; Li, Y.; McKinney, N.; Yoder, D.; Wright, W.; Herrero, H. Comparison of Ground Point Filtering Algorithms for High-Density Point Clouds Collected by Terrestrial LiDAR. *Remote Sens.* **2022**, *14*, 4776. [CrossRef]
30. Boulch, A.; Marlet, R. Fast and robust normal estimation for point clouds with sharp features. *Eurographics Symp. Geom. Process.* **2012**, *31*, 1765–1774. [CrossRef]