EasyIDP: A python package for intermediate data processing in 3D based plant phenotyping

WANG HAOZHOU
University of Tokyo https://orcid.org/0000-0001-6135-402X
Yunlin Duan
Chinese Academy of Agricultural Sciences Institute of Agricultural Resources and Regional Planning
Yun Shi
Chinese Academy of Agricultural Sciences Institute of Agricultural Resources and Regional Planning
Yoichiro Kato
The University of Tokyo Graduate School of Agricultural and Life Sciences Faculty of Agriculture: Tokyo Daigaku Daigakuin Nogaku Seimei Kagaku Kenkyuka Nogakubu
Seish Ninomiya
The University of Tokyo Graduate School of Agricultural and Life Sciences Faculty of Agriculture: Tokyo Daigaku Daigakuin Nogaku Seimei Kagaku Kenkyuka Nogakubu
Wei Guo (guowei@g.ecc.u-tokyo.ac.jp)
The University of Tokyo https://orcid.org/0000-0002-3017-5464

Software

Keywords: Orthomosaic, Photogrammetry, Phenotyping, Reverse calculation, Python

Posted Date: January 5th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-138511/v1

Version of Record: A version of this preprint was published on July 3rd, 2021. See the published version at https://doi.org/10.3390/rs13132622.

SOFTWARE

EasyIDP: A python package for intermediate data processing in 3D based plant phenotyping

Haozhou Wang[1], Yulin Duan[2,3], Yun Shi[2,3], Yoichiro Kato[1,4], Seish Ninomiya[1,5] and Wei Guo[1*]

*Correspondence:
guowei@g.ecc.u-tokyo.ac.jp
[1] International Field Phenomics Research Laboratory, Institute for Sustainable Agro-ecosystem Services, Graduate School of Agricultural and Life Science, The University of Tokyo, 188-0002 Tokyo, Japan
Full list of author information is available at the end of the article

Abstract

Background: The use of 3D based high-throughput phenotyping improves the efficiency of crop management and monitoring practices. The structure-from-motion and multi-view stereo photogrammetry (SfM-MVS) technique, applicable to common RGB digital cameras, has been widely used for this and can be implemented by many commercial and open-source tools. By using such tools, several outputs such as digital orthophoto map (DOM), digital surface model (DSM), and point cloud data (PCD) can be generated. However, there is a gap between these outputs and the final 3D plant phenotyping. For example, calculating plant height and canopy ground cover requires the segmentation of each plot from the whole DOM, DSM, or original image. These intermediate processes are time-consuming, and to the best of our knowledge, there are no easy-to-use alternatives currently available.

Results: In this study, a software package called EasyIDP (easy intermediate data processor) was developed to link the products of SfM-MVS techniques with 3D based plant phenotyping. A lotus (*Nelumbo nucifera*) breeding field was used to demonstrate the following points: 1) clipping (segmenting) SfM-MVS products according to a given plot boundary or region of interest (ROI); 2) transforming the ROI of the SfM-MVS products into high-quality raw images to assist in object detection; and 3) evaluating the accuracy of the previous transformation using manual annotation.

Conclusions: The proposed intermediate data processing tool showed an acceptable accuracy and potential to process the products from SfM-MVS techniques. By using the EasyIDP, a bridge between SfM-MVS products and plant phenotyping was conveniently achieved.

Keywords: Orthomosaic; Photogrammetry; Phenotyping; Reverse calculation; Python

Background

Compared with traditional manual field mensuration, which is time-consuming, labor intensive, and subjective, recently developed 3D reconstruction techniques provide a high-throughput solution for plant phenotyping. The structure-from-motion and multi-view stereo photogrammetry (SfM-MVS) technology, which requires only a common digital camera, has been widely used in both indoor and outdoor applications. For indoor applications, good quality point cloud data (PCD) for individual crops is generated using SfM-MVS [1, 2], and several studies have focused on developing algorithms to classify or segment point clouds to calculate geometric traits [3, 4]. For outdoor experimental field application, although PCD is also an important data source for crop modeling and trait extraction [5, 6, 7], digital orthophoto map (DOM) and digital surface model (DSM) are often required for plot management, to simplify the difficulties encountered when analyzing crops [8, 9, 10]. Aligning the SfM outputs at different times shows the possibilities for time-series analysis [11, 12].

Commercial software such as Pix4DMapper (Pix4D, Lausanne, Switzerland) or Agisoft Metashape (Agisoft LLC, St. Petersburg, Russia) significantly decreases the workload of obtaining the required SfM-MVS outputs from raw images. However, direct application of these outputs to plant phenotyping is not convenient. First, the large file sizes from SfM-MVS outputs mixed with a large amount of background (soil, grass, etc.) in the field, make data analysis and plot extraction more complicated when combined with plants. Hence, one typical data processing method is clipping those outputs according to plot sectors or region of interest (ROI) and making it easier for plot management and this also benefits time-series analyses. Second, practical field applications experience variable and complex environmental conditions, such as light and wind, making it challenging to obtain raw image quality for SfM-MVS products (eg. DOM). For example, Duan et al., [13] reported that 79% of the plots showed overestimation of the ground cover from DOM compared to raw images. Hence, referring to raw images directly to compensate for the loss of DOM or PCD quality would improve the performance of accuracy and time cost.

The objective of this study was to develop an easy-to-use software package to address previously identified difficulties in intermediate data processing between the SfM-MVS outputs and final plant phenotyping results, including: 1) clipping

SfM-MVS outputs to small parts or sectors by a given plot boundary or ROI; 2) transforming the ROI into SfM-MVS products to correspond to the position on original quality raw images; and 3) testing the accuracy and performance of the previous transformation using a case study.

Implementation

The proposed software package EasyIDP (`https://github.com/HowcanoeWang/EasyIDP`) was implemented using the GPL-3.0 license to manipulate SfM-MVS products. Although the source codes were cross-platform owing to the characteristics of the Python language, they were programmed and tested on a Windows 10 64-bit platform and an Intel CPU with math kernel library (MKL). To achieve better package performance, 8 GB RAM and 3.0 GHz CPU are recommended. The general workflow of this tool is shown in Figure 1 and has three main parts: a) 3D reconstruction of the outputs; b) making ROI by other tools for data preparation; and c) EasyIDP functions, corresponding with the 3D reconstruction workflow: (1) SfM stage; (2) MVS stage; and (3) geographic information system (GIS) stage, respectively.

Input data preparation

Image collection and 3D reconstruction

The quality of the raw image is fundamental for the SfM-MVS process as well as the EasyIDP package. It is important that images with balanced brightness, exposure, and minimum motion blur. Overlap of over 50% among each adjacent image is also recommended. Although ground control points (GCPs), manual tie points (MTPs), or even real-time kinematic (RTK) are optional for SfM-MVS software, it is strongly recommended to set several objects, tags, or scale bars to calibrate and define geographic positions. One option is Chilitags (`https://github.com/chili-epfl/chilitags`), as the automatic workflow decreases the workload in the current SfM-MVS software.

After obtaining acceptable quality image data, the SfM-MVS workflow is required. In this study, the commercial Pix4DMapper was used as its default setting to produce the required camera parameters (pmatrix, external, and internal parameters), PCD, and GIS products DOM and DSM. In the future, other commercial soft-

ware such as Agisoft Metashape, or other open-source software will be taken into consideration.

Region of Interest (ROI) making

Currently, the EasyIDP package has no integrated graphical user interface (GUI) for manual ROI marking of all kinds of outputs. Therefore, different external software to mark the ROI on different outputs is required. Please refer user manual (`https://github.com/HowcanoeWang/EasyIDP/wiki`) for detailed operation tutorials.

For the SfM stage, which requires obtaining 2D pixel coordinates on raw images, LabelMe (`https://github.com/wkentaro/labelme`), an image annotation software that produces json annotation files, has been tested and supported. Furthermore, other photo processing software such as GIMP or Photoshop, which displays the pixel coordinates where the cursor hover can be used. These 2D coordinate values where the cursor is hovered can be copied or written to the txt file or csv file for importing into EasyIDP.

For the MVS stage, which produces 3D PCD, 3D coordinates are required. Common point cloud processing software that can pick point cloud points and obtain the 3D coordinates of each point can be used. The open-source CloudCompare (`http://cloudcompare.org/`) has been tested in this project and supports export picked point coordinates to the txt file for later usage. In the future, importing the AutoCAD 3D polygon file (*.dxf) will be considered.

For the GIS stage, which requires obtaining 2D geographic coordinates for the DOM and DSM data, common GIS processing software such as QGIS can be used to produce the required polygon shapefiles (*.shp) as the input for the EasyIDP package.

Package Algorithms

This section demonstrates the calculation algorithm implemented in this package, for specific package documentation please refer to `https://github.com/HowcanoeWang/EasyIDP/wiki`. This section includes: 1) the python packages used for importing several data types and exporting those small sectors clipped from large files; 2) the geometry from the real world to the raw image pixels; 3) camera lens distortion correction; 4) transforming the 2D ROI on one raw image to a corresponding position on another raw image; 5) transforming the 3D ROI on the

point cloud to a corresponding position on other raw images, and 6) transforming the 2.5D ROI on DOM and DSM to corresponding positions on other raw images.

Input and output files

The input file type includes pix4d camera parameters and a pix4d joint rotation-translation matrix (pmatrix) file (*.txt), digital images (*jpeg, *png), point cloud data (*.pcd, *.ply, *.xyz), geotiff files (*.tiff, DOM, and DSM), a polygon shapefile (*.shp), a shapefile geographic projection file (*.prj), and in the future, a 3D polygon file (*.dxf).

Several external Python site packages are used to input these files, all of which are loaded as *numpy.ndarray* based data structure for faster matrix algebra calculations. The *.txt pure text files can be read by the *numpy.loadtxt* module. For *.jpeg and *.png files, the *scikit-image.io.imread* module is used to load the image into *numpy.ndarray* format. For point cloud files, the *open3d.io.read_pcd* module is used, and for Pix4DMapper produced *.ply files, sometimes the color information cannot be correctly loaded, and the *plyfile* package is used to compensate for the color dimension loss. The geotiff file is loaded by the *tifffile* package, including the image layer data and geo-header data. The geo-header data contains the offset, resolution, and geographic projection information, and is used for the transformation between pixel coordinates and geographic coordinates on geotiff images. The *.prj file can be read directly by Python, but the *pyproj* package is used to convert pure text to the Python geo-projection class, and as a projection converter, once the shp projection is not the same as the geotiff projection. For 3D polygon *.dxf file, the *ezdxf* package will be used to convert the *numpy.ndarray* polygon in the future.

After importing all these materials into the EasyIDP package, a clip function is provided to clip both point cloud data and geotiff GIS maps into small sectors. There are two clipping modes: grid clipper and ROI clipper. The grid clipper divides the full DOM into same-sized grids (squares), while the ROI clipper clip the DOM directly based on a given ROI polygon. The grid clipper first generates all the grid boundary polygons, and then views each of them as ROIs to apply the ROI clipper to them. For the clipping of the point cloud, the *open3d.geometry.PointCloud.crop* module is used directly. For clipping the GIS maps, a binary mask with an outlier as a false value is generated based on a given ROI polygon, and the binary mask

is used to clip pixels out of the raw DOM. After that, the left corner offset of the clipped boundary is calculated according to the DOM pixel resolution to obtain the geo-header of the clipped sector, and finally saved to a geotiff file using the *tifffile* package.

Geometry from real world to image pixel

In the geometry between the image and the real world, there are three coordinate systems. The first is the real-world geographic coordinate (xyz_{geo}, unit is m), the second one is the offset camera coordinate (xyz_{cam}), which makes the camera position to the origin (0,0,0) of coordinates and the camera medial axis is used as the z-axis, and the last one is the raw image pixel coordinate (xy_{pix}, unit is pixel).

As shown in Figure 3, assume a point $K(X, Y, Z)$ in real world coordinates (xyz_{geo}) is recorded as k_2 in image raw_2 by the camera. The internal parameters, w and l represent the camera sensor width (horizontal) and length (vertical) in mm, respectively, and f is the camera focal length in mm. For external camera parameters, the camera rotation and position are recorded by $r_i(\omega_i, \varphi_i, \kappa_i)$ [14] and $P_i(Px_i, Py_i, Pz_i)$. The rotation matrix ($\mathbf{R}_i$) of the camera parameters can be calculated by [15], or for Pix4D, can be loaded from the *calibrated_camera_parameters.txt* file:

$$\begin{aligned}\mathbf{R}_i &= \mathbf{R}_{x_i}(\omega_i)\mathbf{R}_{y_i}(\varphi_i)\mathbf{R}_{z_i}(\kappa_i) \\ &= \begin{bmatrix}1 & 0 & 0\\ 0 & cos(\omega_i) & -sin(\omega_i)\\ 0 & sin(\omega_i) & cos(\omega_i)\end{bmatrix}\begin{bmatrix}cos(\varphi_i) & 0 & sin(\varphi_i)\\ 0 & 1 & 0\\ -sin(\varphi_i) & 0 & cos(\varphi_i)\end{bmatrix}\begin{bmatrix}cos(\kappa_i) & -sin(\kappa_i) & 0\\ sin(\kappa_i) & cos(\kappa_i) & 0\\ 0 & 0 & 1\end{bmatrix}\end{aligned}$$

When using the camera location P_i as the coordinate origin (Figure 3.b), the new coordinate K_2' of the real-world point K can be transformed by [15]:

$$K_i' = \begin{bmatrix}X_i'\\ Y_i'\\ Z_i'\end{bmatrix} = \mathbf{R}_i^T \cdot (K - \mathbf{p}_i) = \mathbf{R}_i^T \cdot \left(\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} - \begin{bmatrix}Px_i\\ Py_i\\ Pz_i\end{bmatrix}\right) \quad (1)$$

Projecting K_2' to image coordinate $k_2(x_{k_2}, y_{k_2})$ in [15] (Figure 3.c):

$$\begin{cases} x_{k_i} &= -f \cdot \dfrac{X_i'}{Z_i'} + c_{x_i} \\ y_{k_i} &= -f \cdot \dfrac{Y_i'}{Z_i'} + c_{y_i} \end{cases} \tag{2}$$

where c_x and c_y is the image center in pixel (Figure 3.b), width (horizontal), and length (vertical).

Camera distortion correction

The image shown in Figure 3.c is an ideal undistorted image. However, the lens of the camera often causes distortion, and one extreme example of this distortion is fisheye or a hemispherical camera. Therefore, the pixel coordinate position on an image cannot reflect the true geometric relationship in the real-world. Hence, camera calibration is always required. Commercial software often has built-in camera calibration models, and Pix4D even supports exporting calibrated undistorted images [16].

The transform from the original distorted image pixel coordinates (x_d, y_d) (known) to the corrected undistorted pixel coordinate (x_u, y_u) is described by [15]. First, for Equation 2, let $(\dfrac{X_i'}{Z_i'}, \dfrac{Y_i'}{Z_i'})$ as (x_h, y_h) (unknown):

$$\begin{cases} x_d = -f \cdot x_h + c_{x_i} \\ y_d = -f \cdot y_h + c_{y_i} \end{cases} \Rightarrow \begin{cases} x_h = -\dfrac{(x_d - c_{x_i})}{f} \\ y_h = -\dfrac{(y_d - c_{y_i})}{f} \end{cases}$$

Then, the loading camera model (camera distortion coefficients) from the SfM-MVS software outputs, for Pix4D, the radial distortion (r_1, r_2, r_3) and tangential distortion (t_1, t_2) are in the *calibrated_camera_parameters.txt* file:

$$\begin{cases} \sigma &= x_h^2 + y_h^2 \\ k &= 1 + r_1 \cdot \sigma + r_2 \cdot \sigma^2 + r_3 \cdot \sigma^3 \\ x_{hd} &= k \cdot x_h + t_1 \cdot x_h \cdot y_h + t_2 \cdot (\sigma + 2 \cdot x_h^2) \\ y_{hd} &= k \cdot y_h + t_2 \cdot x_h \cdot y_h + t_1 \cdot (\sigma + 2 \cdot y_h^2) \end{cases}$$

The (x_{hd}, y_{hd}) is the corrected result for $(\dfrac{X_i'}{Z_i'}, \dfrac{Y_i'}{Z_i'})$ in Equation 2:

$$\begin{cases} x_u &= -f \cdot x_{hd} + c_{x_i} \\ y_u &= -f \cdot y_{hd} + c_{y_i} \end{cases}$$

Transforming the ROI from the raw image to another raw image

The geometry of transforming ROI from one image to another is shown in Figure 3.a. Transforming a polygon can be simplified to a repeatable polygon corner point transformation. Assuming a point K (value unknown) in the real-world coordinate (xyz_{geo}) is recorded as k_1 and k_2 on image raw_1 and raw_2 by the camera, respectively, the raw_1 is where the ROI is marked while the (raw_2) is the target image we want this ROI transformed to, and the k_1 (value known) is one ROI polygon corner. As the projection of the 3D real world point to a 2D pixel point ($K \xrightarrow{(1)} K_1' \xrightarrow{(2)} k_1$) is a dimension reduction step with Z information loss, revert transform from the given pixel coordinate $k_1(x_{k_1}, y_{k_1})$ back to the real world K_r ($k_1 \xrightarrow{(2)^{-1}} K_1' \xrightarrow{(1)^{-1}} k_r$) can only obtain the equation with Z_1' as parametric:

$$K_r = \left(\mathbf{R}_1^T\right)^{-1} \cdot \begin{bmatrix} X_1' \\ Y_1' \\ Z_1' \end{bmatrix} + P_1 = \left(\mathbf{R}_1^T\right)^{-1} \cdot Z_1' \begin{bmatrix} \frac{c_{x_1} - x_{k_1}}{f} \\ \frac{c_{y_2} - y_{k_1}}{f} \\ 1 \end{bmatrix} + P_1$$

This means that without specifying the exact value of Z_1', the previous step will produce a line rather than a specific point in the real world as well as on the other raw image. The ideal and most accurate method is marking the same ROI on two images (get k_1 and k_2 known), and then calculating their intersection to obtain the exact Z value of K_r, followed by $K_r \xrightarrow{(1)} K_3' \xrightarrow{(2)} k_3$ to obtain the pixel coordinate on the third image. In this study, the height value of the camera was used as Z_1' directly, and reasonably, our case study pre-experiment result (Figure S3, Additional file 3) showed that this assumption still needs some improvements to make the accuracy acceptable.

Transform ROI from the point cloud to the raw images

For point cloud data, the Pix4D software will produce PCD with XYZ values in geographic coordinates ($X_{real}, Y_{real}, Z_{real}$). Hence, the ROI coordinate ($X_{roi}, Y_{roi}, Z_{roi}$) marked in Pix4D produced point cloud could be linked to the

real-world coordinate $(X_{real}, Y_{real}, Z_{real})$ directly ($XYZ_{real} = XYZ_{roi}$). However, sometimes software produces an offset point cloud with a *offset.xyz* file, and $XYZ_{real} = XYZ_{roi} - XYZ_{offset}$.

After obtaining 3D coordinates in the real-world, the 3D (xyz_{geo}) to 2D (xy_{pix}) calculation can be conducted using the previous $K \xrightarrow{(1)} K_i' \xrightarrow{(2)} k_i$ pipeline, or if perspective lens cameras are used (not fisheye or spherical camera), the joint rotation-translation matrix (pmatrix, 3 rows 4 columns), which are calculated by Pix4D, and can be directly used to achieve the same result [17]:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = PMatrix_{3\times4} \cdot \begin{bmatrix} X_{real} \\ Y_{real} \\ Z_{real} \\ 1 \end{bmatrix}$$

Transforming the ROI from the GIS map to raw images

The ROI of GIS coordinates is often stored as a shapefile polygon (*.shp). However, sometimes the projection of the Shapefile is not the same as the GIS product (DOM and DSM) projection. The projection coordinate transform is provided by EasyIDP to solve this inconsistency. Meanwhile, the shapefile polygon only contains XY geographic information (xy_{geo}), a transformation from the geographic coordinate to the pixel coordinate of the DSM is required to obtain the height (Z) value of the ROI on the related DSM pixel (z_{geo}). After obtaining the full set of geographic coordinates (xyz_{geo}), the same procedure as for the previous 3D to 2D section could be operated.

Case Study: Lotus field

Field data and image collection

To obtain the materials for package development and testing, an experimental lotus (*Nelumbo nucifera*) field in Nishi-Tokyo, Tokyo, Japan was used as a case study (Figure 2.a). A total of 112 different varieties of lotus were sown in squire cultivation plots (Figure 2.b). Ordinary local management practices were used to manage all trials.

The DJI Inspire 1 drone (DJI, China) with onboard camera Zenmuse X5 Pro (DJI, China) was used to acquire images with a flight height of 30 m. The flight

plan was designed using a double-grid style with the LitchAPP (VC Technology Ltd. UK) software, with >90% overlap of the pictures with the front and sides. The Pix4Dmapper Pro (Pix4D S.A., Switzerland) was used to obtain the SfM-MVS products. The parameters for all SfM-MVS steps in the software were used as the defaults, and the GCPs were measured using the Hemisphere RTK differential GNSS devices (Hemisphere GNSS)(Figure 2.c). The device for 3D reconstruction processing includes Intel Xeon E5-2690 v4 @2.60GHz CPU, 128 GB RAM, two NVIDIA GeForce GTX1080Ti (Driver: 26.21.14.3186) GPUs, and Windows 10 Pro, 64-bit operating system. The details of the digital data collected and produced are shown in Table 1.

After processing all flights and obtaining DOM and DSM, the QGIS was used to mark and export plot boundary polygons and their IDs on the map (Figure 2.b).

Z value determination

In this case study, after clipping out the ROI of each plot, two z values were calculated. The first one was the 5^{th} percentile mean z value, $mean(Z_{p5})$, that equals the mean z values of all the points smaller than the threshold of the 5^{th} percentile, used for catching the elevation (height) of the water surface to define the z value of the ROI. The other was the 95^{th} percentile mean z value, $mean(Z_{p95})$, which is equal to the mean of all point z values, and is larger than the threshold of the 95^{th} percentile, used for catching the top of the lotus leaves. Furthermore, a ground region was selected manually (the middle corridor in Figure 2.b where scale bars were placed), the mean Z value of this region, $mean(Z)_{ground}$, was used as the ground elevation. The lotus height was equal to $mean(Z_{p95}) - mean(Z)_{ground}$.

Performance evaluation

To determine how the transformation performs and what contributes to the deviation, the expected transformation results were made using LabelMe annotation software manually and used to compare with that calculated by the EasyIDP package. Three indicators, intersection of union (IoU) performance criterion [18], precision, and recall, were used to evaluate the similarities between the package output and manual marking, and could be calculated by (refer to Tresch et al., [9] Figure 3 for the IoU diagram of each area):

$$
\begin{aligned}
IoU &= \frac{\text{intersection area}}{\text{union area}} \\
precision &= \frac{\text{intersection area}}{\text{program area}} \\
recall &= \frac{\text{intersection area}}{\text{manual area}}
\end{aligned}
$$

Two kinds of comparison were involved. For the first one, three plots with different lotus densities (N3E6: sparsest, S2W4: medium sparse, N2W5: densest) were selected, and all related raw images were marked manually. The pixel Euclidean distance from the IoU center to the image center (c_x, c_y) was also calculated, and the relationship between the indicator values and Euclidean distance were simply discussed. Second, for each plot, the smallest IoU Euclidean distance raw image was selected to mark the manual reference, and the overall trend of the indicators was simply analyzed.

Results and Discussion

As mentioned in the introduction, intermediate data processing contains two main tasks: 1) clipping large outputs to small sectors according to the variable file types of the grids or ROI (Application 1); 2) transforming the ROI of each output to the corresponding position on high-quality raw images (Application 2), the accuracy evaluation is also included. Additionally, some potential applications of this tool for data annotation and augmentation in deep learning are discussed.

Application 1: Clipping SfM outputs

For the grid clipper, a demo result is shown in Figure 4.a, using the Windows 10 file management system. These sector images still contain geographic information that can be overlaid directly with the original DOM image (Figure 4.b), the grid clipper supports the clip of the GIS shapefile polygon (*.shp) over each grid (Figure 4.c), while in Figure 4.d, with a view of each grid, all the annotations belonging to the current grid could be summarized together.

For the ROI clipper, Figure 5.a-b, shows an example of DOM clipping along a given ROI, while for Figure 5.c-d, shows an example of PCD clipping. In this study, the output files did not occupy too much storage space (Table 1) because the SfM-MVS processing quality was limited. The seldom used background for plant phenotyping analysis still occupys at least 60% of the area (Figure 2.b) by clipping

the ROI to individual geotiff files with plot names, a large amount of storage space could be saved, and this helped the plot-based time-series management and data analysis.

The time-series analysis could be conducted after collecting the continuous data and clipping into sectors by ROI. A simple example (plot S2E1) for time-series height tracking is shown in Figure 6. By adding RTK and GCPs in the corner of the plot (Figure 2.c), the image XY plane was calibrated to the same position using the timeline, reflected by the cultivation plot edges being linked together without significant deviation (Figure 6 first row).

With respect to the Z-axis, although the height of the lotus showed a general trend of increasing, the times 20170531, 20170612, and 20170718 showed an abnormal trend and their heights were significantly greater than the later days. This should be affected by the quality of the SfM-MVS processing. By visual checking, on these days, the 3D reconstructed ground was found to have slightly leaned from west to east, meaning that the $mean(Z)_{ground}$ was not the actual ground height near that plot and caused the abnormality in height. Furthermore, the z value (elevation) of each time was not calibrated in the same manner as the XY plane; hence, the $mean(Z)_{ground}$ varied differently along the timeline, the deviation could reach 0.3 m, which was unacceptable compared with the crop height. In the future, the GCPs for the z-axis should also be taken into consideration; for example, a visible box or pole could be set in the field. How to evaluate the quality of the 3D reconstruction and the adjustment for the 3D reconstruction workflow to fit the characteristics of the agriculture should also be considered.

Application 2: Transformation outputs ROI to raw images

Although the SfM-MVS software provides options for the quality (density) of the MVS stage, the highest quality could provide an ultra-high resolution of the outputs. However, this processing is unacceptable computationally intensive, and time-consuming for high-throughput agriculture; meanwhile, a study showed that the ground coverage estimated from DOM is greater than that from the raw images, which took over 79% of the plots because of the ghosting effects caused by the wind and cloud [13]. For 2D image-based trait analysis, for example, ground coverage, leaf projection area, or object detection, marking ROI on fast produced low-quality

DOM and linking them back to raw images, could save a considerable amount of time without losing accuracy. This technique was called ROI transformation in this study and in previous investigations reverse calculation [9, 13]. The demo of the ROI transformation from the DOM and PCD on some of the raw images is shown in Figure 5.e.

The results of the performance evaluation are shown in Figure 7. For plots N2W5 and S2W4, most of the IoU values were over 90%, surprisingly, for the simplest and sparse plot N3E6, although the IoU values were greater than 75%, the performance was moderate. This is because of the automatic z value calculation of ROI (elevation, $mean(Z_{p5})$). For manual marking, the reference was the edge of the plot, which means that the manual ROI elevation was the plot edge (the broken blue line in the second row of Figure 7), while the elevation auto-selected is the red solid line in Figure 7. The plot N3E6 had almost 25 cm differences between the two elevations, mainly caused by the transparent water. The 3D reconstruction built the bed mud rather than the water surface for this plot. This shows that the ROI height selection is of great importance for the transformation accuracy. Another trend in this figure was that the IoU decreased with the distance from the ROI center to the photo center. This means that the position of ROI on the raw images (the view angle where taking raw photo) also reflected the transform accuracy.

Figure 8 shows how the ROI elevation selection and ROI positions on the raw images affect transformation accuracy. Three different heights (bottom $mean(Z_{p5})$, middle $mean(Z)$, and top $mean(Z_{p95})$) were selected as ROI elevations respectively (Figure 8.a), the deviation variation on the different raw images is shown. The closer to the raw image center, the fewer the effects of the height selection, which was highly related to the view angle. Figure 7 also shows once the pixel distance to the image center was smaller than 800 (size of image is 4608×3456 pixels), the moderate performance plot N3E6 could achieve an IoU of greater than 90%. Hence, to minimize the effects of the unsuitable ROI elevation selection, raw images where the ROI locates in the center as a priority are recommended. Based on this finding, for all 112 plots, the minimum distance raw image for each plot was selected and the expected ROI results were marked manually, the distribution of the accuracy indicators was shown in Figure 9, the peaks are around 98% while the minimum value was still greater than 90%. Considering the difficulties in manual marking in

some plots where all corners were covered in leaves and nonidentical (See Additional file 1, Figure S1), the ROI transformation accuracy has been determined to be acceptable.

The EasyIDP package also provides a computer vision-based distortion correction method (Additional file 2, Figure S2), the condition for using this algorithm is strictly limited to the flat region. In the lotus case study, it worked as expected only on a plot with sparse and flat leaves laid on the water surface. For lotus leaves that are stretched out on the water surface for more than 20 cm, the matching results no longer follow the plot edge; instead, they follow the movement of the top leaf pattern in different angles of the view, but they also have a high possibility of mismatching.

Potential application for deep learning

Applying deep learning frameworks to common digital images for object classification or segmentation is now a popular area of agricultural research [19, 20, 21]. However, limited by the huge memory consumption during convolution, the input image size is often limited to small sizes ($< 1{,}000 \times 1{,}000$), while the original DOM size is hundreds of times larger ($> 10{,}000 \times 10{,}000$). Although common GIS software (e.g., ArcGIS or QGIS) provides toolboxes for clipping a whole DOM to small grids (parts) and may also provide programmable API for the script or batch processing, from our practical experiences, the processing steps as well as the API learning cost, build a wall beyond widespread use. In this EasyIDP package, a grid clipper was provided to solve this requirement using only a few lines of code (Figure 4.a-b).

How to transform the GIS format (*.shp) to deep learning required an annotation format (*.json) after clipping, and this was another challenge. The previous clipping method provides a solution that results in large DOM as a common small-size digital camera image, and the small objects within each grid could be marked directly on the produced grid sectors. For those objects whose size is greater than the grids, for example, the plot boundary, the grid clipper also proved a convenient way to break down (intersect) and integrate them with the grid boundary (Figure 4.c-d). All the annotations beyond this grid could be automatically transferred to the LabelMe supported json file and could be imported directly into the deep learning framework.

The value of high-quality raw images has often been ignored in recent studies. The SfM step builds a geometric relationship between raw images and GIS products, PCD and DOM, and provides potential solutions for two major problems in deep learning: 1) unstructured point data analysis and 2) high-efficiency training data augmentation.

First, applying a deep learning framework on unstructured PCD is significantly different from that for structured matrix image data. Although the point cloud deep learning framework exists (e.g., PointNet [22]) and some studies applying the adapted deep learning framework to PCD obtained by LiDAR directly [23, 24, 25], Van de Zedde et al., [26] showed a method of applying common image deep learning on raw images and projecting results to PCD using the geometry relationship, which changes the difficult unstructured data application to a common structured image application.

Second, most deep learning frameworks require at least thousands of training images to avoid overfitting and ensure the versatility of the model, but annotating many training data sets is labor intensive. Data augmentation was proposed to decrease the workload of annotation by linear transformations such as rotating, zooming, or flipping on annotated images, and applied in some agricultural deep learning studies [19, 27]. However, this image processing-based data augmentation cannot exceed the photo collected from different view angles in the real world. Hence, Beck et al.,[28] proposed an automatic indoor robotic annotation system to collect natural augmented crop training data in agricultural systems. However, to the best of our knowledge, there is no annotation data collection system for open fields. The raw images collected for 3D reconstruction naturally contained an abundant view angle (rotation in 3D), height (scale zoom), and environmental conditions (light, cloud shadow, wind shake, soil color of different wetness), which is good material for data augmentation. However, the difficulty in identifying each plant and its links with annotations means that there is a great potential, which is being ignored, and the problem could easily be solved by using the ROI transformation function in the EasyIDP package. As shown in Figure 10, the original DOM was clipped by 500px×500px grids, and the grid "x6-y7" which contains several lotus flowers was chosen to make the flower annotations using the LabelMe software. Then, the annotation json file was loaded in EasyIDP and the reverse calculation was applied

to transform these annotations onto raw images automatically. As discussed before, when the annotations were closer to the centers of the images, fewer deviations were obtained. The last two columns showed that the cases with the farthest distances had the worst performances. A distance of less than 1000 px was acceptable for automatic annotation marking.

Future prospects

The EasyIDP package is currently only a pre-release as it is still under construction. There are many aspects of the program that could be further modified and developed, such as: 1) all the functions mentioned previously were only developed and tested using the products of Pix4Dmapper, while Agisoft Metashape, another well-known commercial software program, and other open source programs, have not yet been adapted; 2) as mentioned in the implementation section, the SfM stage also provides the possibility to convert the ROI from one raw image to another raw image, a pre-experiment (Additional file 3, Figure S3) showed that using the geographic height of the image (z value of p_1) as the unknown Z_1' involved great deviation and unacceptably low levels of accuracy, and thus this requires improvement in the future by reverting the PMatrix or distortion correction; 3) although adding Z axis control points helps in the height calibrations, integrating terrestrial cameras as well as aerial drones also improves the height accuracy. How to mix two different camera types and routes to perform 3D reconstruction and fitting EasyIDP under these circumstances still needs to be investigated; 4) compared with perspective lens camera, which have a limited sight angle, the hemispherical (fisheye) and spherical cameras could record full surrounding information with one shot, which would greatly improve the image data collection efficiency. How to use a severely distorted camera for 3D reconstruction and apply full EasyIDP functions on them should be taken into consideration.

Conclusions

Commercial or open-source software significantly decreases the workload of 3D reconstruction when using SfM-MVS products from raw images for 3D based high-throughput phenotyping. However, there are still some intermediate data processing steps required to apply these SfM-MVS products. To our knowledge, there is no easy-to-use tool for this task currently available. In this study, a Python package

called EasyIDP was proposed to build a bridge from SfM-MVS outputs to plant phenotyping and provide services for: 1) clipping large SfM-MVS outputs to acceptable file types and sizes following the plot boundary or given ROI; 2) linking ROI on those outputs to corresponding positions with original resolution raw images. The accuracy of the previous transformation has been validated with manually marked references, and the impact factors for the accuracy have been discussed. By using this tool, not only could the quality of the image-based trait analysis such as ground cover or leaf area be improved, but it also shows the great potential for using automatic augmented deep learning training with data. In the future, several extension studies include adapting more SfM-MVS software programs, improving deviation corrections, and integrating mixed types of cameras as well as spherical lens cameras should be developed.

Availability and requirements

- **Project name:** EasyIDP
- **Project home page:** https://github.com/HowcanoeWang/EasyIDP
- **Project documentation** https://github.com/HowcanoeWang/EasyIDP/wiki
- **Operating system(s):** Using the source codes as Python package is platform independent (windows 10 (x64) and Intel CPU with math kernel library (MKL) support are tested and recommended).
- **Programming language:** Python
- **Other requirements:** For using source code as package: Python 3.7 or higher, numpy 1.18.1 or higher, scikit-image 0.16.2 or higher; opencv-python 3.4.2.16 or higher, pyproj 2.6.1.post1, pyparsing 2.0.1 or higher are required, and following pure Python packages were included in the source code without installation: pyshp, Send2Trash, ezdxf, and plyfile.
- **License:** GPL-3.0
- **Any restrictions to use by non-academics:** Free to use for any purpose, forever.

Acknowledgments

We would like to thank the technique staff of Institute for Sustainable Agro-ecosystem Services (ISAS) for the lotus field management.

Authors' contributions

HW and WG conceived the ideas and designed the methodology; WG collected UAV images and obtained the outputs of SfM-MVS, HW programmed the package, analyzed the data, and YD and YS contributed to the DOM clipping and annotation transformation. YK and SN supervised this study. All authors discussed and wrote the manuscript and gave final approval for publication.

Funding

This study was partially funded by the JST CREST Program JPMJCR16O2, JPMJCR1601, JPMJCR1603, JPMJCR1512, SICORP Program JPMJSC16H2, aXis program JPMJAS2018, and National Natural Science Foundation of China U19A2061.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The download link of the example data, and Jupyter notebook codes for drawing all results figures, and the LaTeX codes of this manuscript, are available on `https://github.com/HowcanoeWang/EasyIDP.paper`.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] International Field Phenomics Research Laboratory, Institute for Sustainable Agro-ecosystem Services, Graduate School of Agricultural and Life Science, The University of Tokyo, 188-0002 Tokyo, Japan. [2] Key Laboratory of Agricultural Remote Sensing, Ministry of Agriculture, 100081, Beijing, China. [3] Institute of Agricultural Resources and Regional Planning, Chinese Academy of Agricultural Sciences, 100081, Beijing, China. [4] International Rice Research Institute, Metro Manila, the Philippines. [5] Plant Phenomics Research Center, Jiangsu Collaborative Innovation Center for Modern Crop Production,Nanjing Agricultural University, 210095, Nanjing, China.

References

1. Wang Y, Wen W, Wu S, Wang C, Yu Z, Guo X, et al. Maize Plant Phenotyping: Comparing 3D Laser Scanning, Multi-View Stereo Reconstruction, and 3D Digitizing Estimates. Remote Sensing. 2019 Jan;11(1):63.
2. Rossi R, Leolini C, Costafreda-Aumedes S, Leolini L, Bindi M, Zaldei A, et al. Performances Evaluation of a Low-Cost Platform for High-Resolution Plant Phenotyping. Sensors. 2020 Jun;20(11):3150.
3. Ziamtsov I, Navlakha S. Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds. Plant Physiology. 2019 Dec;181(4):1425–1440.
4. Artzet S, Chen TW, Chopard J, Brichet N, Mielewczik M, Cohen-Boulakia S, et al. Phenomenal: An Automatic Open Source Library for 3D Shoot Architecture Reconstruction and Analysis for Image-Based Plant Phenotyping. bioRxiv. 2019 Oct;p. 805739.
5. Jay S, Rabatel G, Hadoux X, Moura D, Gorretta N. In-Field Crop Row Phenotyping from 3D Modeling Performed Using Structure from Motion. Computers and Electronics in Agriculture. 2015 Jan;110:70–77.
6. Sun S, Li C, Chee PW, Paterson AH, Jiang Y, Xu R, et al. Three-Dimensional Photogrammetric Mapping of Cotton Bolls in Situ Based on Point Cloud Segmentation and Clustering. ISPRS Journal of Photogrammetry and Remote Sensing. 2020 Feb;160:195–207.
7. Zhu B, Liu F, Xie Z, Guo Y, Li B, Ma Y. Quantification of Light Interception within Image-Based 3D Reconstruction of Sole and Intercropped Canopies over the Entire Growth Season. Annals of Botany. 2020 Mar;p. mcaa046.
8. Sun S, Li C, Paterson AH, Jiang Y, Xu R, Robertson JS, et al. In-Field High Throughput Phenotyping and Cotton Plant Growth Analysis Using LiDAR. Frontiers in Plant Science. 2018 Jan;9:16.
9. Tresch L, Mu Y, Itoh A, Kaga A, Taguchi K, Hirafuji M, et al. Easy MPE: Extraction of Quality Microplot Images for UAV-Based High-Throughput Field Phenotyping. Plant Phenomics. 2019 Nov;2019:1–9.

10. Chen CJ, Zhang Z. GRID: A Python Package for Field Plot Phenotyping Using Aerial Images. Remote Sensing. 2020 May;12(11):1697.
11. Dong J, Burnham JG, Boots B, Rains G, Dellaert F. 4D Crop Monitoring: Spatio-Temporal Reconstruction for Agriculture. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, Singapore: IEEE; 2017. p. 3878–3885.
12. Han L, Yang G, Yang H, Xu B, Li Z, Yang X. Clustering Field-Based Maize Phenotyping of Plant-Height Growth and Canopy Spectral Dynamics Using a UAV Remote-Sensing Approach. Frontiers in Plant Science. 2018 Nov;9:1638.
13. Duan T, Zheng B, Guo W, Ninomiya S, Guo Y, Chapman SC. Comparison of Ground Cover Estimates from Experiment Plots in Cotton, Sorghum and Sugarcane Based on Images and Ortho-Mosaics Captured by UAV. Functional Plant Biology. 2017;44(1):169.
14. Pix4D support. Yaw, Pitch, Roll and Omega, Phi, Kappa Angles; 2020. https://support.pix4d.com/hc/en-us/articles/202558969-Yaw-Pitch-Roll-and-Omega-Phi-Kappa-angles.
15. Pix4D support. How Are the Internal and External Camera Parameters Defined?; 2020. https://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined.
16. Pix4D support. Menu Process > Save Undistorted Images; 2020. https://support.pix4d.com/hc/en-us/articles/202557929.
17. Pix4D support. What Does the Output Params Folder Contain?; 2020. https://support.pix4d.com/hc/en-us/articles/202977149.
18. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision. 2010 Jun;88(2):303–338.
19. Zhou C, Ye H, Yu g, Hu J, Xu Z. A Fast Extraction Method of Broccoli Phenotype Based on Machine Vision and Deep Learning. Smart Agriculture. 2020;2(1):121.
20. Feng A, Zhou J, Vories E, Sudduth KA. Evaluation of Cotton Emergence Using UAV-Based Imagery and Deep Learning. Computers and Electronics in Agriculture. 2020 Oct;177:105711.
21. Desai SV, Balasubramanian VN, Fukatsu T, Ninomiya S, Guo W. Automatic Estimation of Heading Date of Paddy Rice Using Deep Learning. Plant Methods. 2019 Dec;15(1):76.
22. Qi CR, Su H, Mo K, Guibas LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:161200593 [cs]. 2017 Apr;.
23. Jin S, Su Y, Gao S, Wu F, Hu T, Liu J, et al. Deep Learning: Individual Maize Segmentation From Terrestrial Lidar Data Using Faster R-CNN and Regional Growth Algorithms. Frontiers in Plant Science. 2018 Jun;9:866.
24. Jin S, Su Y, Wu F, Pang S, Gao S, Hu T, et al. Stem–Leaf Segmentation and Phenotypic Trait Extraction of Individual Maize Using Terrestrial LiDAR Data. IEEE Transactions on Geoscience and Remote Sensing. 2019 Mar;57(3):1336–1346.
25. Jin S, Su Y, Song S, Xu K, Hu T, Yang Q, et al. Non-Destructive Estimation of Field Maize Biomass Using Terrestrial Lidar: An Evaluation from Plot Level to Individual Leaf Level. Plant Methods. 2020 Dec;16(1):69.
26. Van de Zedde R, Kootstra G, Shi W, Jiang H. Plant-Part Segmentation Using Deep Learning and Multi-View Vision. Biosystems Engineering. 2019 Sep;187:81–95.
27. Han J, Shi L, Yang Q, Huang K, Zha Y, Yu J. Real-Time Detection of Rice Phenology through Convolutional Neural Network Using Handheld Camera Images. Precision Agriculture. 2020 Jun;.
28. Beck MA, Liu CY, Bidinosti CP, Henry CJ, Godee CM, Ajmani M. An Embedded System for the Automated Generation of Labeled Plant Images to Enable Machine Learning Applications in Agriculture. arXiv:200601228 [cs]. 2020 Jun;.

Additional Files

Additional file 1: Figure S1

Difficult examples for manually marking the expected transformation results. The plot corner or edge is partly or mostly covered by the lotus leaves and cannot be identified.

Additional file 2: Figure S2

Deviation correction by template matching. The basic assumptions for this correction include: 1) the package calculated transform position was not too far from the true value; 2) the region near the ROI was flat, which means that the pattern of the ROI object in the different images (raw images) would not change significantly from the different angles of the view. Based on the previous assumptions, to avoid the similarity of the neighboring plant mismatching, the square sector of the ROI on the original image (DOM in this case, Figure S3.a) and square sector of the transformed package result with 30% external buffer (Figure S3.b) were clipped out to match rather than matching directly on the original images. A pre-experiment using feature matching (Figure S3.c) failed, the agricultural applications involved a quantity of similarity key points, the feature matching always mismatched, and could not obtain acceptable invariance for both rotation and scale. To solve the rotation and scale invariance, the transform matrix from the ROI polygon points in Figure S3.a and ROI polygon points in Figure S3.b were calculated using the *opencv.findHomography* function, and this transformation was applied to the clipped original image sector (Figure S3.a) to Figure S3.d, which shares the same scale and rotation with the target image sector (Figure S3.b). The transformed new image was used as the template (Figure S3.e) matching pixel by pixel (*scikit-image.feature.match_template*). The highest similarity position was used for the deviation corrected result (Figure S3.f-g).

Additional file 3: Figure S3

The pre-experiment results of transforming ROI from raw images to other raw images, assuming the camera height as missing Z values. (a) is the image marked by the ROI. (b)–(e) show some of the ROI transform results.

Table 1 Trail field and image collection information

Flight date (yyyymmdd)	**No. of raw images**	**Size of raw images** (GB)	**Size of DOM** (MB)	**Size of DSM** (MB)	**Size of PCD** (MB)	**Processing time** (min)
20170525	266	1.64	38.5	34.6	89.2	39.7
20170531	151	0.94	40.4	37.6	60.0	47.3
20170603	141	0.87	46.5	34.1	60.3	18.8
20170612	285	1.74	38.1	32.9	93.8	101.5
20170616	136	0.84	39.1	33.4	60.1	11.1
20170703	138	0.88	36.3	33.2	58.5	11.0
20170707	132	0.82	40.7	32.4	57.1	10.0
20170711	138	0.86	41.8	33.4	57.8	14.4
20170718	135	0.84	38.5	35.4	56.9	10.1
20170724	142	0.90	34.5	32.9	59.2	10.3

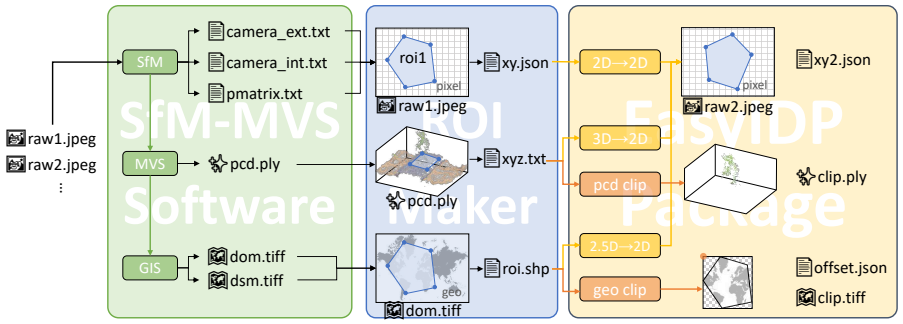

Figure 1 The workflow of the proposed method. After obtaining the image data, the 3D reconstruction (SfM-MVS) software was used to generate the outputs of each processing step. Then, the region of interest (ROI) was marked via external commercial or open-source tools. Finally, the EasyIDP package was used to deal with and connect different SfM-MVS products and ROI, including reverse calculations of the projected position of the ROI and clipping the output of the ROI ranges.

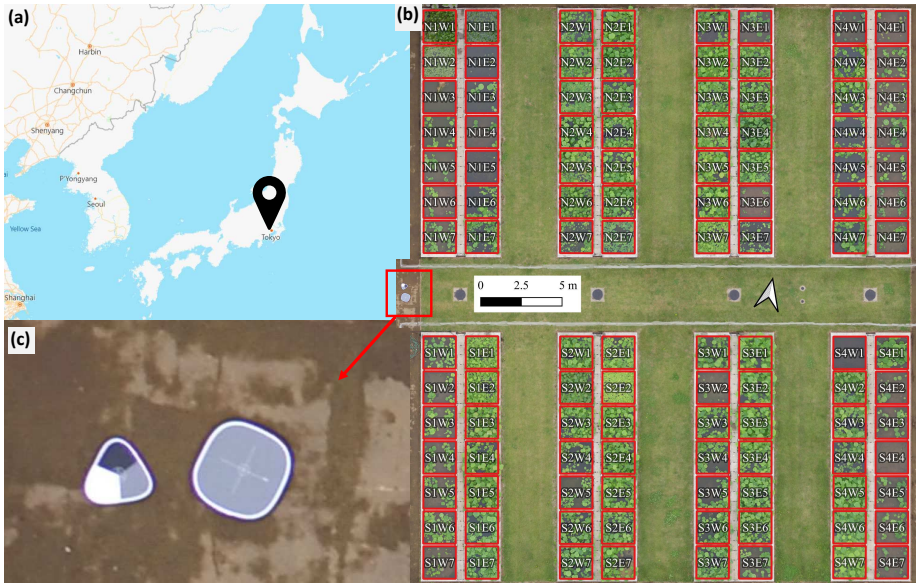

Figure 2 The experimental lotus plot condition. (**a**) The experimental plot was located in Nishi-Tokyo, Tokyo, Japan; (**b**) The labels of the cultivation plots, the orthomosaic (DOM) shown here, was collected on May 5, 2017. The labels and boundaries of the plots were made using QGIS software and saved to a Shapefile (*.shp) for later usage; (**c**) shows the device used for obtaining the geographic position and functioning as a ground control point to link different flight time series together

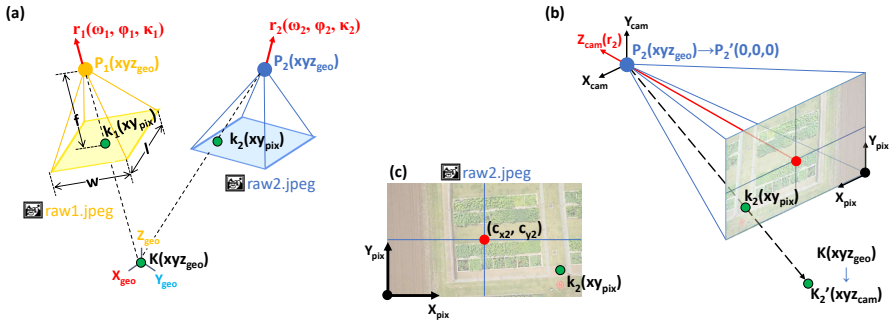

Figure 3 The geometry of the camera model and 3D reconstruction. There were three different coordinate systems involved: the real-world coordinate system (xyz_{geo}, **a**), camera coordinate system (xyz_{cam}, **b**), and image coordinate system (xy_{pix}, **c** in pixels). (**a**) shows how a point in the real world was recorded in different raw images, r_i represents the camera rotation of the raw image i; P_i represents the camera position of the image i; k_i represents the 2D pixel coordinates on the image i; w is the sensor width (mm); f is the sensor focal length, and l is the sensor length (mm). (**b**) offsets the camera position to (0,0,0) to better link the pixel coordinate positions (k_i) with the real-world position, where (c_{x_i}, c_{y_i}) is the image pixel center.

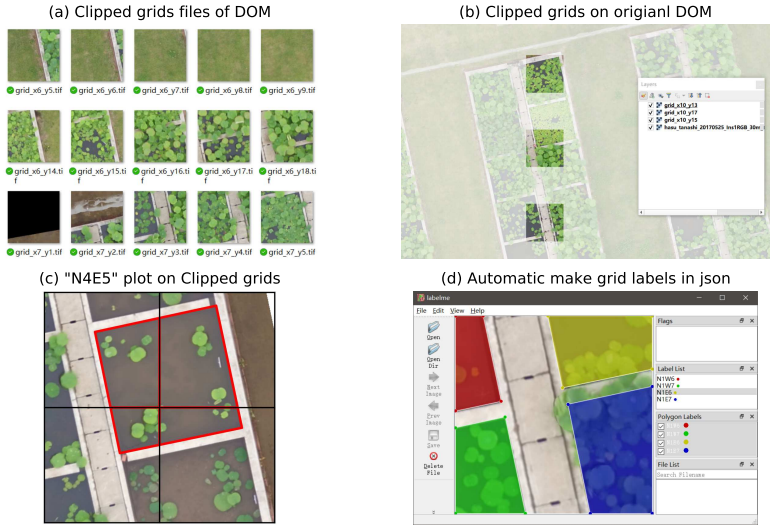

Figure 4 The example for the digital orthophoto map (DOM) grid clipping by given size. (**a**) shows the output geotiff files of the clipped girds in a Windows 10 file management system. (**b**) showed that the saved geotiff files contained geographic information that could be overlaid in the same position as the original DOM in the QGIS interface. (**c**) shows the clipping of a given ROI according to each grid. (**d**) for each grid, different ROI sectors could be summarized and saved to the annotation json file (readable into LabelMe annotation software)

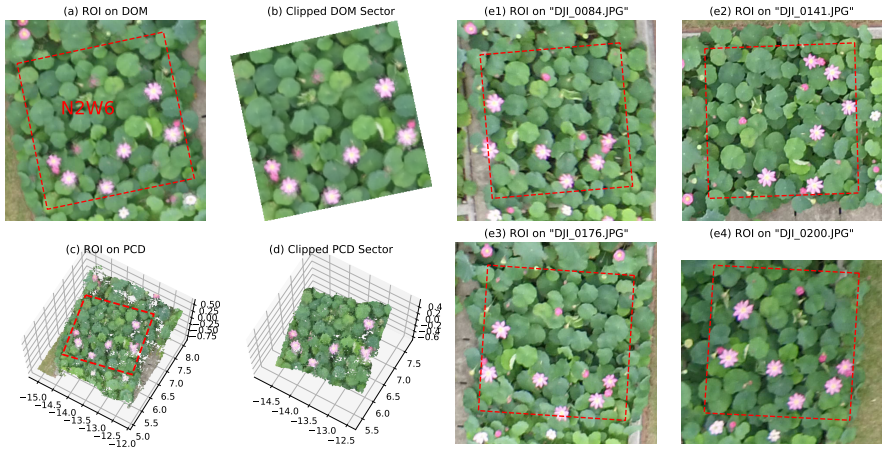

Figure 5 The example for region of interest (ROI) transformation between different digital orthophoto map (DOM), point cloud data (PCD) and raw images. (**a**) shows the ROI marked by QGIS and is displayed on the DOM. (**b**) shows the clipped sector of the DOM by the ROI. (**c**) shows the ROI on the PCD and (**d**) is the PCD clipped by the ROI. (**e**) shows the ROI transformation results on the raw images.

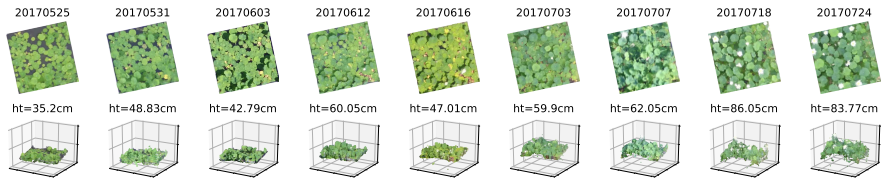

Figure 6 The time-series tracking of the ROI, S2E1 plot as example, ranging from May 25 to July 24, 2017.

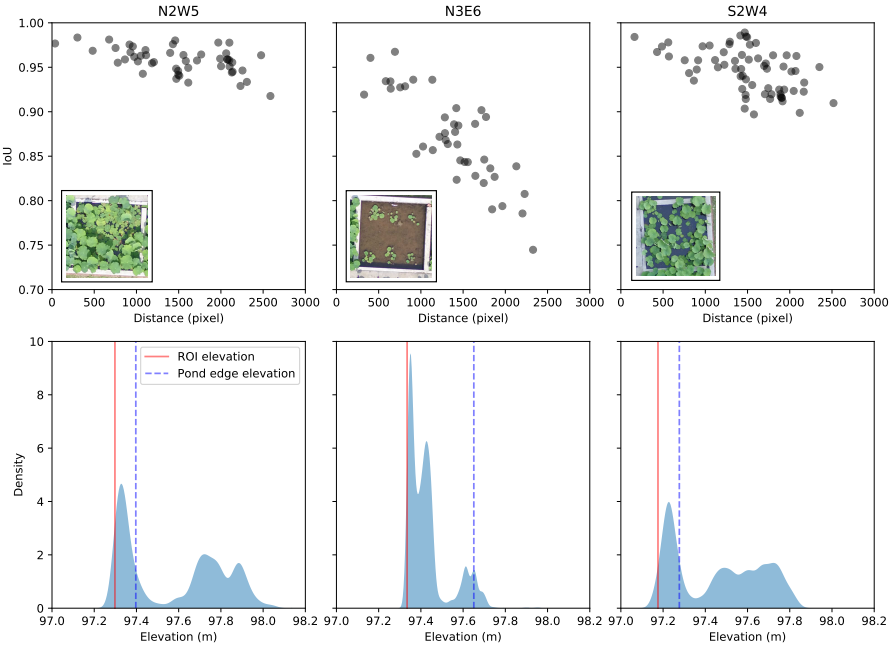

Figure 7 The transformation accuracy examines the results from the completely manually marked plots. Three plots with different lotus leaf densities and heights were selected. All the expected ROI transformation positions were marked manually on raw images using the LabelMe annotation software. The distance is the Euclidean pixel distance from the intersection of union (IoU) center to the photo center. The distribution of each pixel point height is shown in the blue area, the height of the ROI (red solid lines) is the mean value of all the pixel points smaller than the 5^{th} percentile threshold, and the height of the plot edge (blue broken lines) is the mean value of a random 10 points picked from QGIS on the DSM.

Figure 8 The effects of the ROI position and ROI height for the transformation. (**a**) shows three different height choices of ROI (5% percentile, mean, 95% percentile) by point cloud display. (**b**) and (**c**) shows two different ROI positions on the raw images and related ROI transformation results.

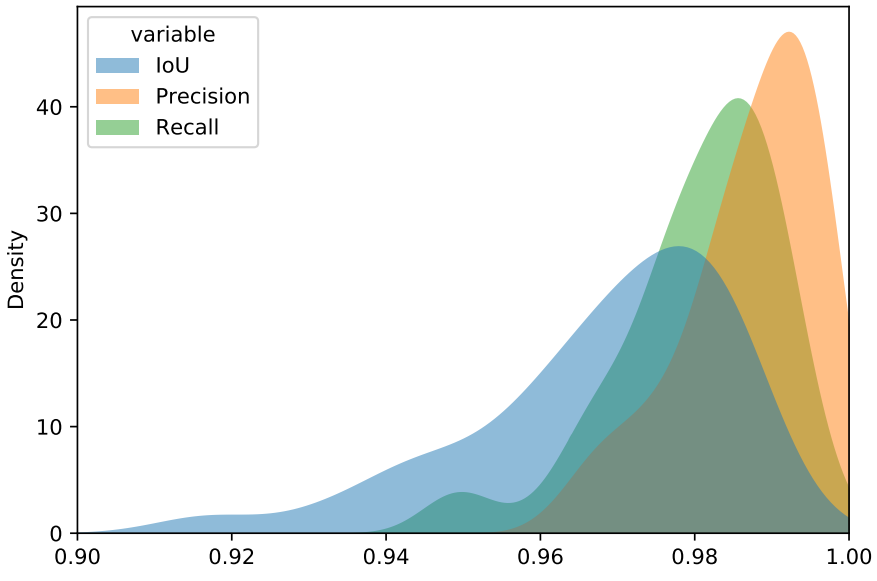

Figure 9 The distribution of the transformation accuracy indicators for the central manual marked for all 112 lotus plots. For each plot, only raw images with the smallest Euclidean pixel distance from the ROI to the image center were selected and manually marked.

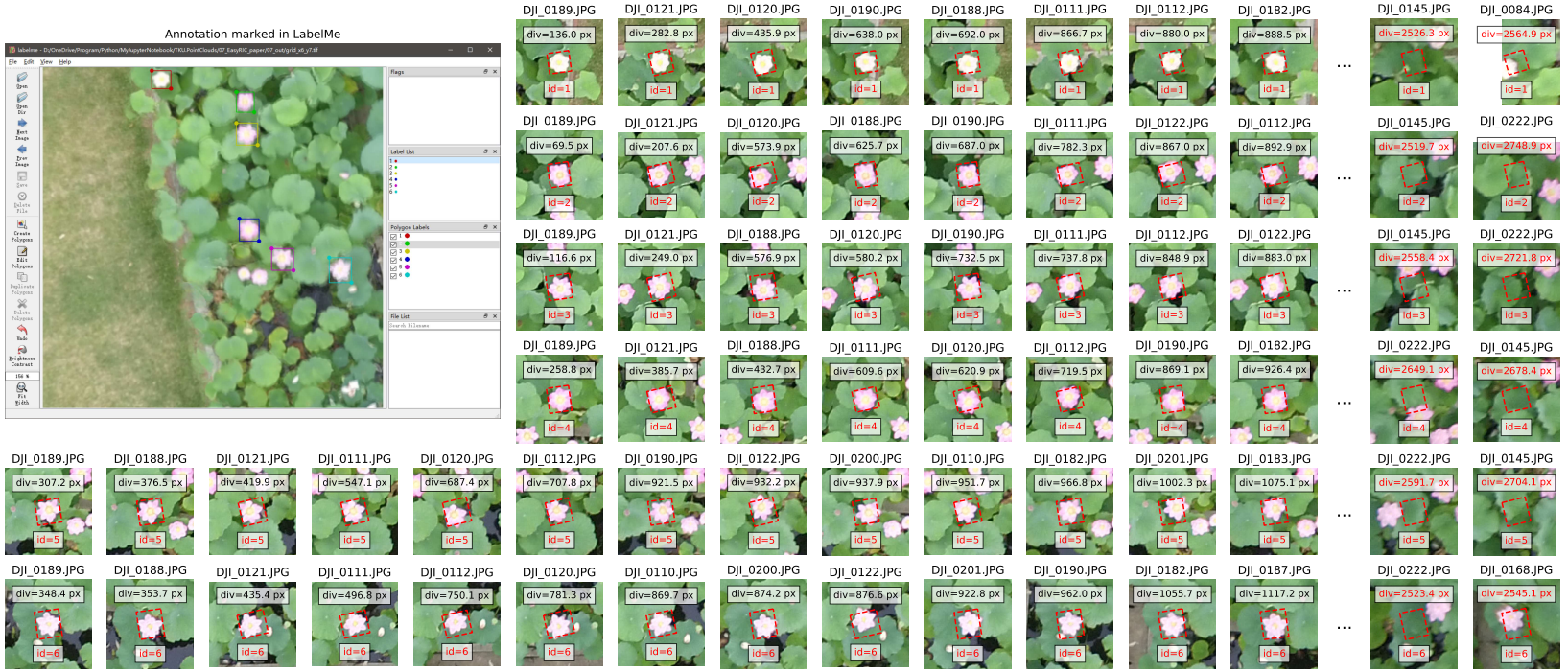

Figure 10 Potential use for data annotation and augmentation in deep learning. First, the DOM was clipped to 500px×500px grids, the grid "x6-y7" was marked with 6 annotation rectangles for lotus flowers using LabelMe. The annotation json file was imported to EasyIDP and reverse calculations were performed on all raw images. The "div" means the distance from the annotation center to the image center, smaller means closer to the image center

Figures

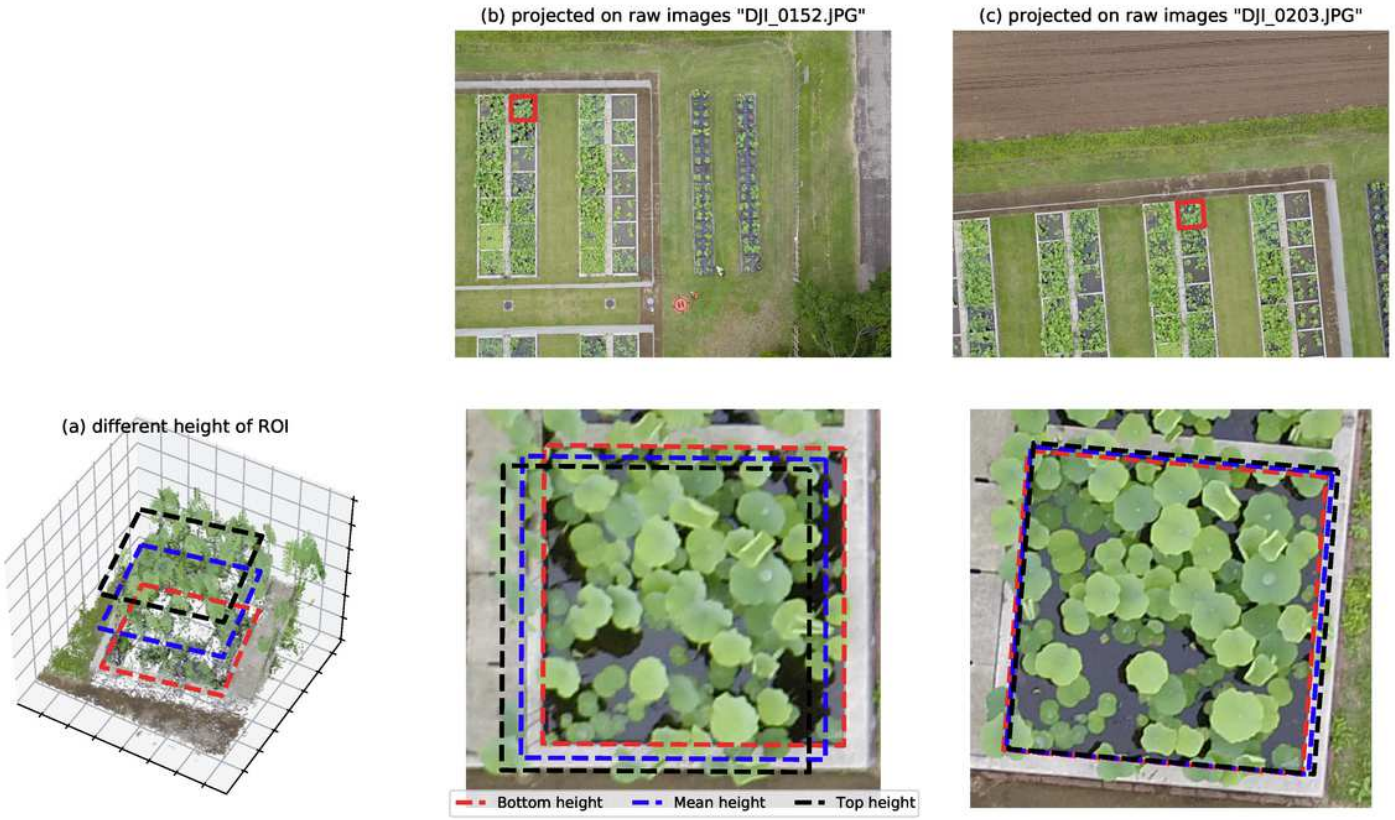

Figure 1

The workflow of the proposed method. After obtaining the image data, the 3D reconstruction (SfM-MVS) software was used to generate the outputs of each processing step. Then, the region of interest (ROI) was marked via external commercial or open-source tools. Finally, the EasyIDP package was used to deal with and connect different SfM-MVS products and ROI, including reverse calculations of the projected position of the ROI and clipping the output of the ROI ranges.

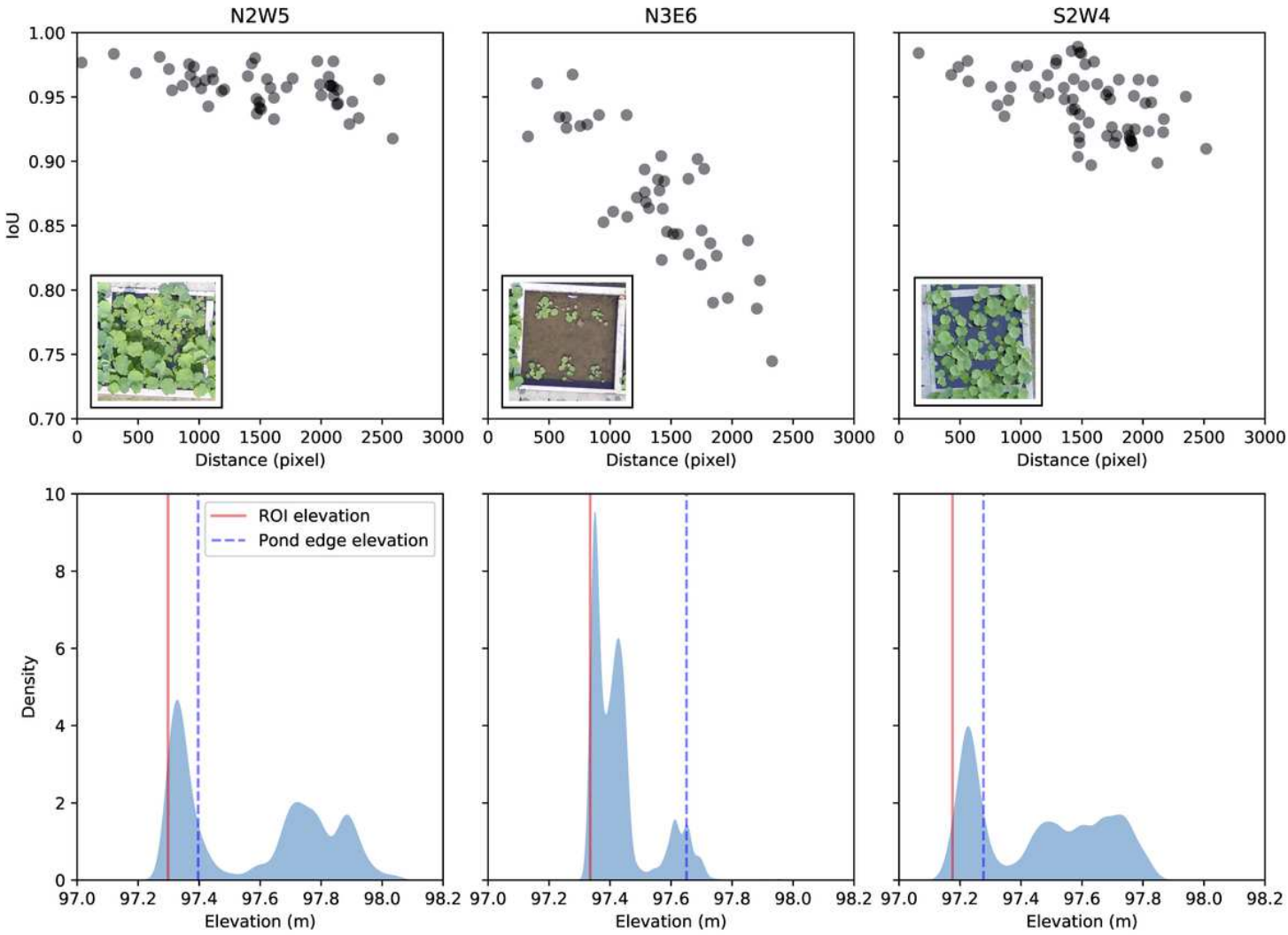

Figure 2

The experimental lotus plot condition. (a) The experimental plot was located in Nishi-Tokyo, Tokyo, Japan; (b) The labels of the cultivation plots, the orthomosaic (DOM) shown here, was collected on May 5, 2017. The labels and boundaries of the plots were made using QGIS software and saved to a Shapefile (*.shp) for later usage; (c) shows the device used for obtaining the geographic position and functioning as a ground control point to link different flight time series together

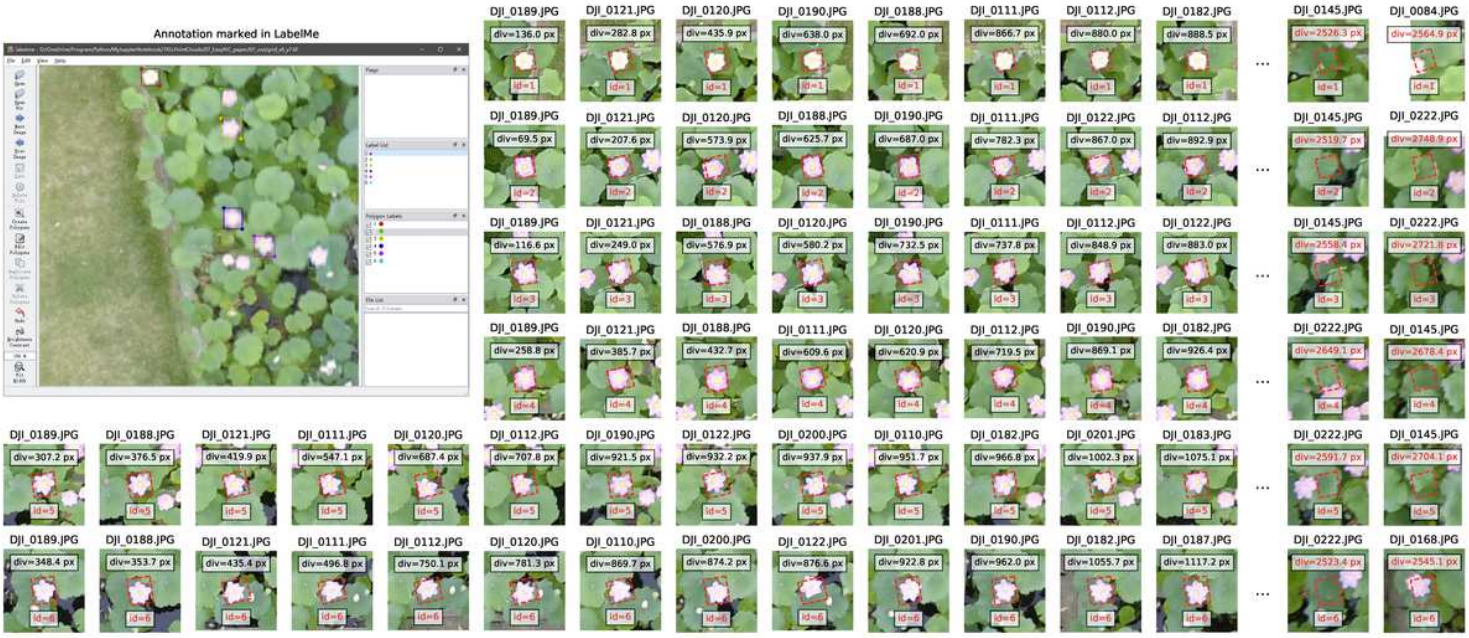

Figure 3

The geometry of the camera model and 3D reconstruction. There were three different coordinate systems involved: the real-world coordinate system (xyzgeo, a), camera coordinate system (xyzcam, b), and image coordinate system (xypix, c in pixels). (a) shows how a point in the real world was recorded in different raw images, ri represents the camera rotation of the raw image i; Pi represents the camera position of the image i; ki represents the 2D pixel coordinates on the image i; w is the sensor width (mm); f is the sensor focal length, and l is the sensor length (mm). (b) offsets the camera position to (0,0,0) to better link the pixel coordinate positions (ki) with the real-world position, where (cxi , cyi) is the image pixel center.

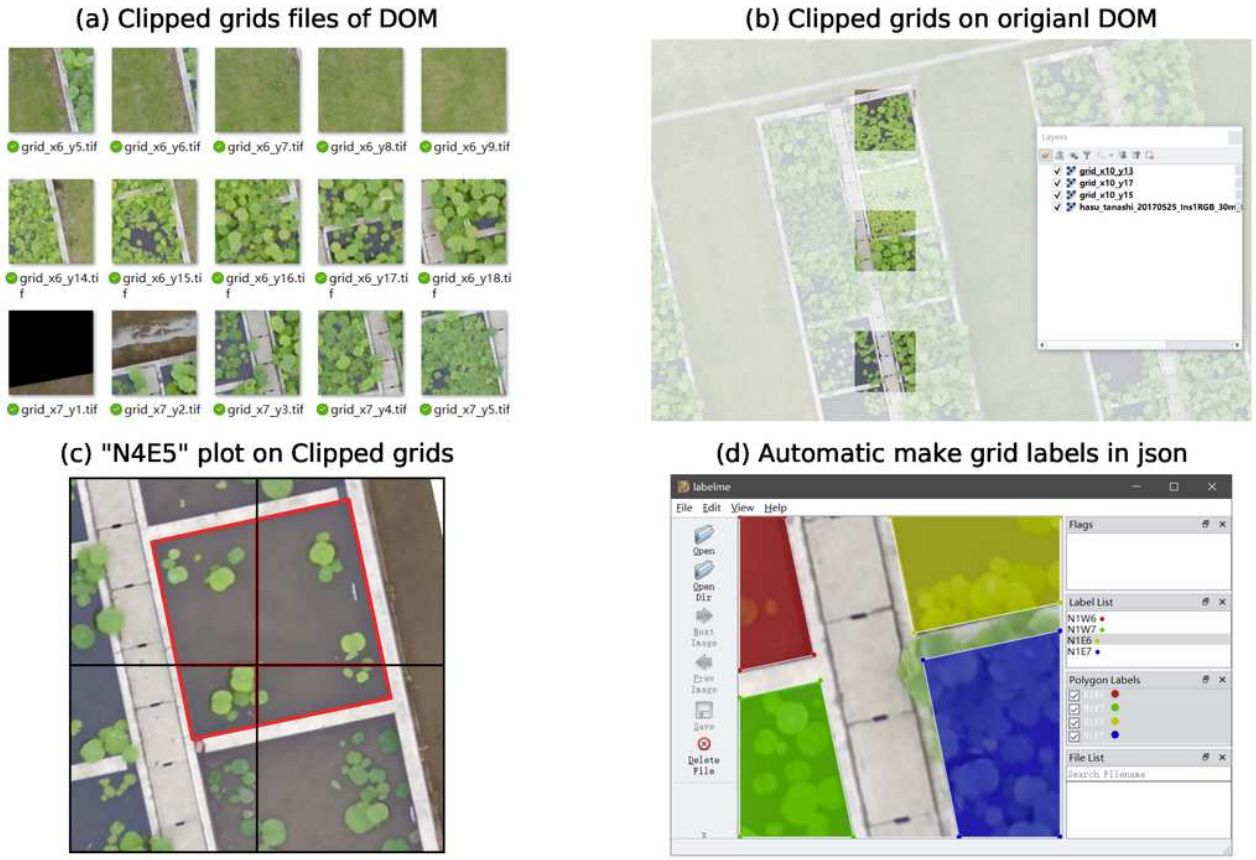

Figure 4

The example for the digital orthophoto map (DOM) grid clipping by given size. (a) shows the output geotiff files of the clipped girds in a Windows 10 file management system. (b) showed that the saved geotiff files contained geographic information that could be overlaid in the same position as the original DOM in the QGIS interface. (c) shows the clipping of a given ROI according to each grid. (d) for each grid, different ROI sectors could be summarized and saved to the annotation json file (readable into LabelMe annotation software)

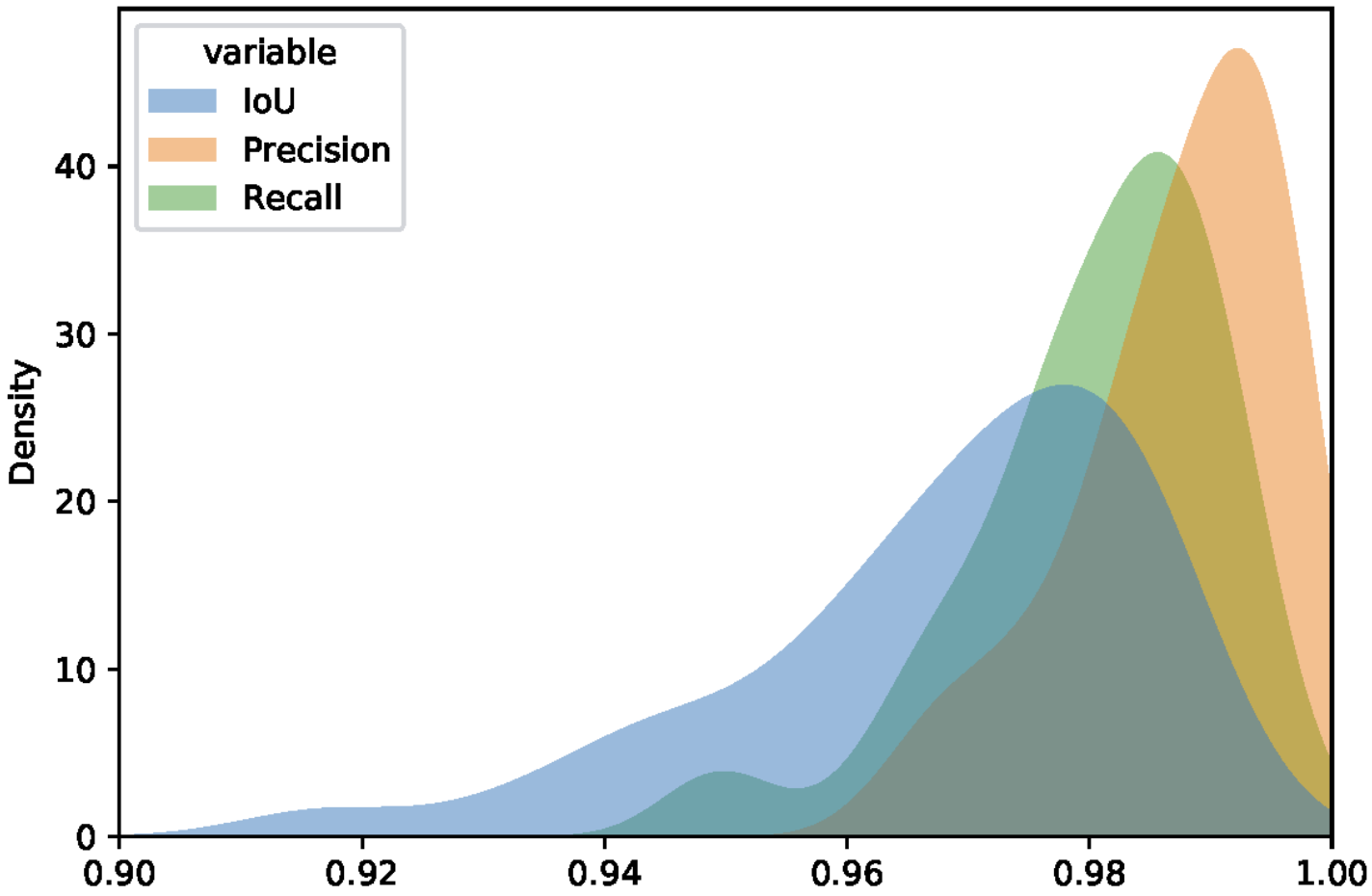

Figure 5

The example for region of interest (ROI) transformation between different digital orthophoto map (DOM), point cloud data (PCD) and raw images. (a) shows the ROI marked by QGIS and is displayed on the DOM. (b) shows the clipped sector of the DOM by the ROI. (c) shows the ROI on the PCD and (d) is the PCD clipped by the ROI. (e) shows the ROI transformation results on the raw images.

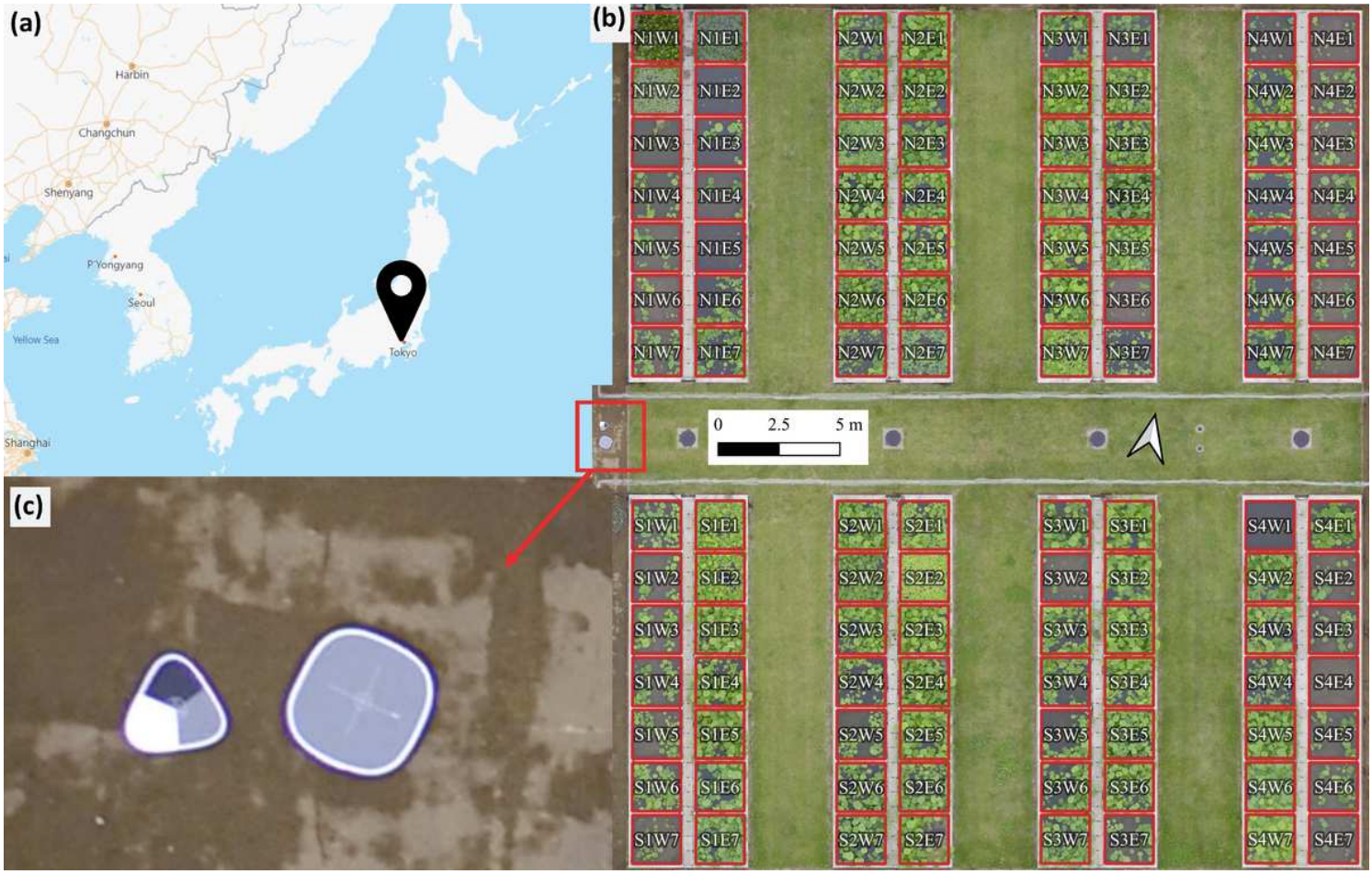

Figure 6

The time-series tracking of the ROI, S2E1 plot as example, ranging from May 25 to July 24, 2017. Note: The designations employed and the presentation of the material on this map do not imply the expression of any opinion whatsoever on the part of Research Square concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. This map has been provided by the authors.

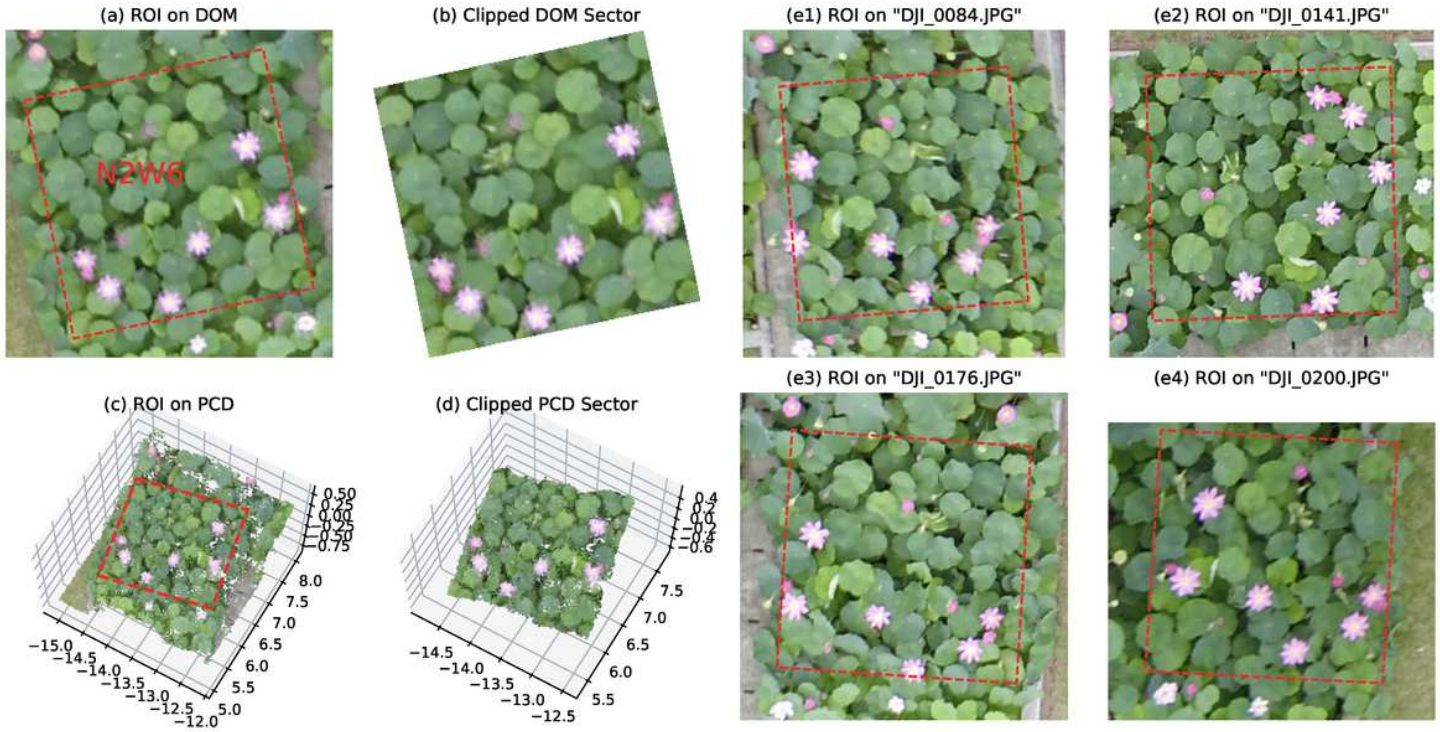

Figure 7

The transformation accuracy examines the results from the completely manually marked plots. Three plots with different lotus leaf densities and heights were selected. All the expected ROI transformation positions were marked manually on raw images using the LabelMe annotation software. The distance is the Euclidean pixel distance from the intersection of union (IoU) center to the photo center. The distribution of each pixel point height is shown in the blue area, the height of the ROI (red solid lines) is the mean value of all the pixel points smaller than the 5th percentile threshold, and the height of the plot edge (blue broken lines) is the mean value of a random 10 points picked from QGIS on the DSM.

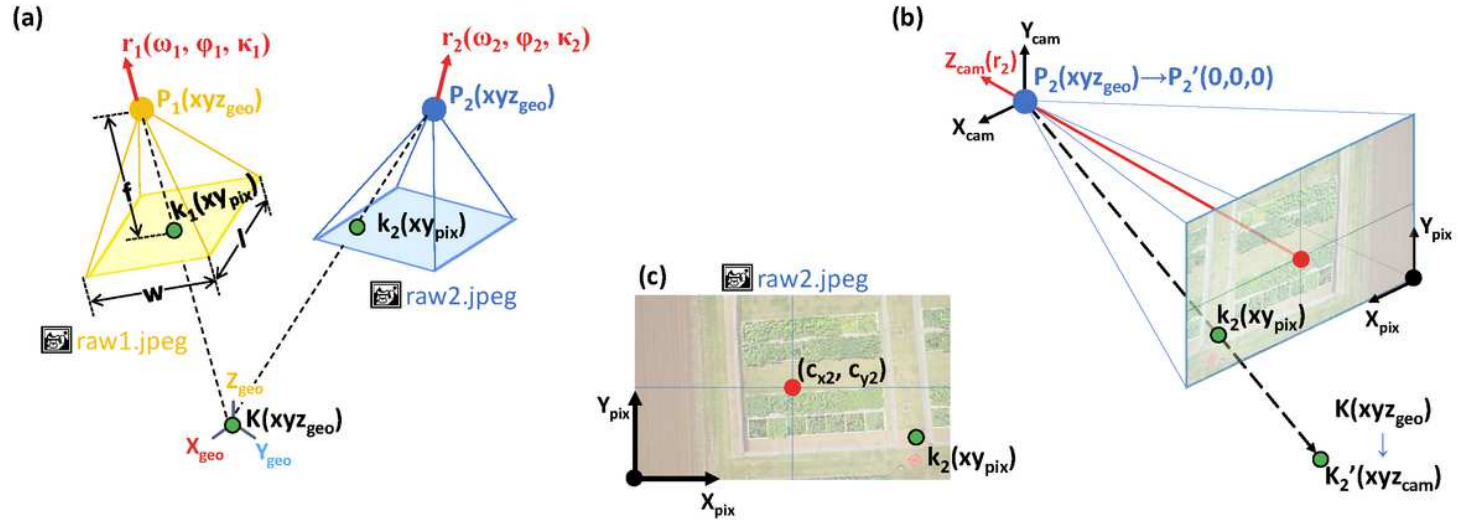

Figure 8

The effects of the ROI position and ROI height for the transformation. (a) shows three different height choices of ROI (5% percentile, mean, 95% percentile) by point cloud display. (b) and (c) shows two different ROI positions on the raw images and related ROI transformation results.

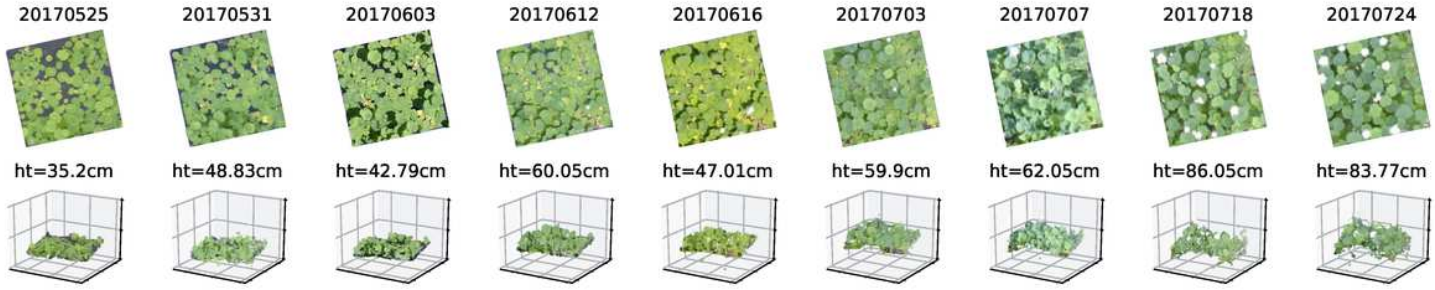

Figure 9

The distribution of the transformation accuracy indicators for the central manual marked for all 112 lotus plots. For each plot, only raw images with the smallest Euclidean pixel distance from the ROI to the image center were selected and manually marked.

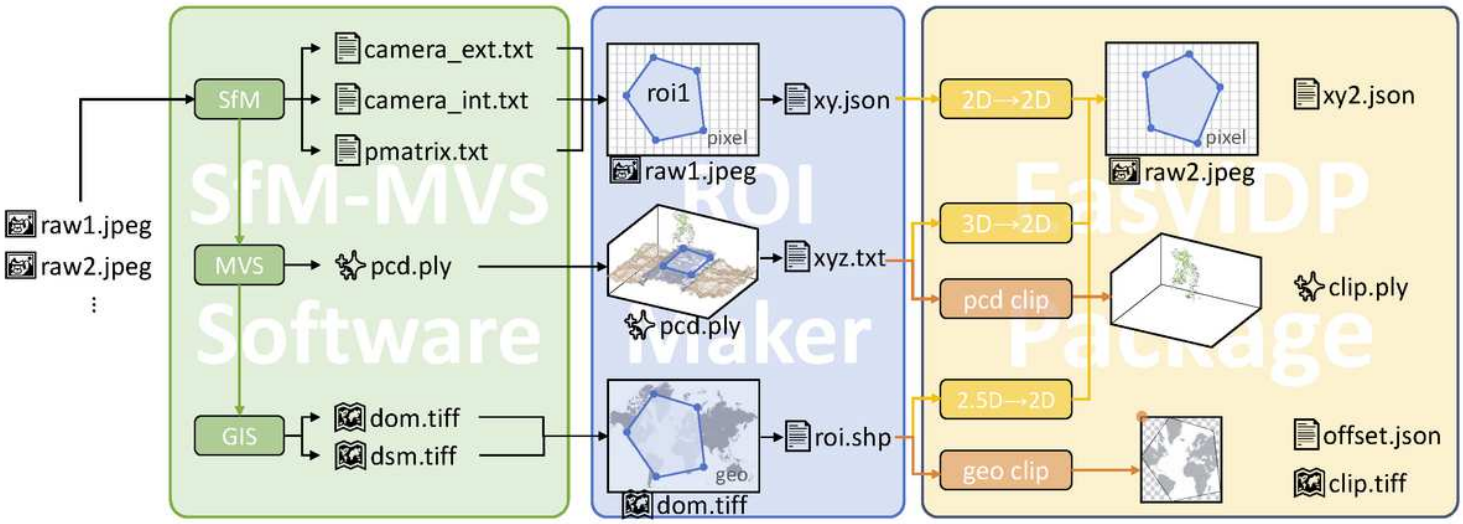

Figure 10

Potential use for data annotation and augmentation in deep learning. First, the DOM was clipped to 500px×500px grids, the grid ”x6-y7” was marked with 6 annotation rectangles for lotus flowers using LabelMe. The annotation json file was imported to EasyIDP and reverse calculations were performed on all raw images. The ”div” means the distance from the annotation center to the image center, smaller means closer to the image center

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Additionalfile1.pdf
- Additionalfile2.pdf
- Additionalfile3.pdf
- EasyRICReverseCalculation.pdf