EasyIDP v2.0: An Intermediate Data Processing Package for Photogrammetry-Based Plant Phenotyping

Haozhou Wang[0000−0001−6135−402X] and
Wei Guo[0000−0002−3017−5464]

Abstract Photogrammetry techniques (3D reconstruction) has been commonly used in applications of high-throughput plant phenotyping. However, some intermediate processing still requires manual operation, which is often time-consuming and labor-intensive. For example, only some Regions of Interest (ROIs) rather than the entire plot or field are worth analyzing, and manually cropping them from images, geotiffs, and point clouds necessitates alternating use of different software programs. In addition, Since the orthomosaic is stitched from different image views, it is easily affected by wind, causing ghost effects and seamline distortion, the raw images are clearer and more competitive for image analysis , but matching the corresponding regions from raw images to actual field positions is challenging. Thus, to decrease the workload required for the intermediate data processing, we improved our previous EasyIDP package for several commonly used solutions, including (1) cropping and saving ROIs from the entire field for raw images, geotiffs (orthomosaic), and point clouds; (2) finding the corresponding ROI from a 3D model to raw 2D images (backward projection), and returning image analysis results back to the 3D model (forward projection). By using this package, researchers are able to crop ROIs from digital orthomosaic (DOM), improve ROI image quality, and map deep learning results on raw images to real-world positions, with just a few lines of code.
Keywords photogrammetry, reverse calculation, unmanned aerial vehicle, drone, 3D reconstruction

Haozhou Wang
Graduate School of Agricultural and Life Sciences, The University of Tokyo, Japan, e-mail: haozhou-wang@g.ecc.u-tokyo.ac.jp

Wei Guo
Graduate School of Agricultural and Life Sciences, The University of Tokyo, Japan, e-mail: guowei@g.ecc.u-tokyo.ac.jp

1 Introduction

By 2050, global population levels are expected to reach 9.6 billion, with a further rise to 10.9 billion by the end of the 21st century. (Gerland et al., 2014); however, the current rate of yield gains needs to be doubled to meet food security requirements (Ray et al., 2013). To achieve this requirement, smart farming is being implemented, incorporating advanced technologies such as plant phenotyping, remote sensing, and artificial intelligence., has recently received considerable attention from researchers, farmers, and governments (Pivoto et al., 2018). Compared to conventional, time-consuming, and labor-intensive manual approaches, plant phenotyping has shown great potential and efficiency in measuring plant traits and monitoring growth conditions (Yang and Xu, 2021). Additionally, by integrating with advanced 3D reconstruction techniques like 3D light detection and ranging (LiDAR) and photogrammetry, it has become more cost-efficient to estimate advanced 3D geometric traits like leaf area and canopy volume (Su et al., 2018; Liu et al., 2021), which is often not so cost-efficient through conventional approaches.

As already summaried in previous manuscript about EasyIDP v1.0 (Wang et al., 2021), the general plant phenotyping pipeline including the following steps: plot design, image data collection, 3D reconstruction, intermediate processing and traits calculation. In the stage of plot design and image data collection, proper control point design (Oats et al., 2019) is crucial for image alignment and ensuring output consistency in time-series analysis. Properly planning the flight route to achieve sufficient overlap is also vital for the quality of 3D reconstruction. Additionally, configuring camera settings such as ISO, exposure, and focal length is important for maintaining high raw image quality, which significantly impacts the final outputs (Guo et al., 2021b).

Then for the 3D reconstruction step, there are several techniques available such as LiDAR and depth camera (Paulus, 2019; Okura, 2022). Photogrammetry, a subset of 3D reconstruction also referred to as structure from motion and multi-view stereo (SfM-MVS) necessitates merely a standard consumer-grade digital RGB camera. Its cost-effectiveness has led to its extensive application in plant phenotyping researches(Herrero-Huerta et al., 2023; Gao et al., 2021; Zermas et al., 2020). It can produce not only plant 3D architectural models in the formats of a point cloud or mesh but also field 2D maps (digital orthomosaic maps, DOM) and elevation maps (digital surface model, DSM) with geographical information (e.g., longitude and latitude) from aerial images. These data are fundamental components for plant phenotyping analysis (Guo et al., 2021b).

Intermediate data processing (preprocessing) often involves preparing the plant data in the proper format for later analysis. It commonly includes the cropping of regions of interest (ROIs) and the bi-directional linking between 3D models (often geographical coordinates for aerial surveys) and raw images (pixel coordinates), with separate directions called backward and forward projecting, respectively.

Currently, to conduct intermediate processing involving the cropping of ROIs, researchers need to switch between different tools and software tailored to various input data types (GeoTiff, Point Cloud, and raw images). This process is time-consuming and prone to human errors, especially when handling a large batch of files. Although Metashape (Agisoft LLC, St. Petersburg, Russia; `https://www.agisoft.com`) Pro version offers visualization functions for backward and forward projection demands, users are unable to export and utilize them for further analysis. Additionally, to our knowledge, there are no available tools to perform these projection operations, let alone a comprehensive tool integrating ROI cropping, backward and forward projection for plant phenotyping researchers. Specifically:

First, there are no easy-to-use tools for ROI cropping. Agronomists and plant breeders frequently focus on particular segments of the entire field, known as the regions of interest (ROIs) (Mortensen et al., 2019). These ROIs separate plots or subplots subjected to varied treatments, or they could specify areas of plant organs such as leaves of lettuce, heads of sorghum and broccoli. Extracting these regions from the entire field makes them fit well with the plot design . It is also suitable for data management. Currently, such an operation is often manually conducted using geographic information system (GIS) software, such as ArcGIS (Esri, Redlands, CA, USA, commercial software) and QGIS (`www.qgis.org`, free software). It would be time-consuming to manually process a large field's time-series high-throughput phenotyping data. Although most GIS software provides batch processing APIs, it requires professional programming and well-trained GIS skills. Thus, some researchers have also developed their tools to simplify this operation, including EasyMPE (Tresch et al., 2019), drone-dataflow (Mortensen et al., 2019), and GRID (Chen and Zhang, 2020). However, these tools lack detailed documentation and software maintenance, and drone-dataflow (Mortensen et al., 2019) depends on commercial software Matlab (MathWorks Inc, Natick, MA, USA), which is still difficult and unaffordable for agronomists and breeders without a computer science or geoscience background.

Second, there are no easy-to-use tools for backward projecting that convert positions on the 3D model or the real world to those on the raw images. Due to the complex natural conditions in the open field, the generated field ortho-maps after photogrammetry often suffer from double mapping (ghost effect) and seamline distortion (Lin et al., 2021). Some studies have tried to fix this poor-quality issue using machine learning algorithms (Hu et al., 2019, 2021; Velumani et al., 2021) or multispectral sensors (Guo et al., 2021a; Lu et al., 2022). However their approaches can be time-consuming, computationally costly, and lack robustness on different crops. Little attention has been paid to the raw imageswhich often have better image quality than the photogrammetry-produced field maps. The drawback of raw images is not having geographical information for each pixel, making it very challenging to match plants from images to their corresponding field locations accurately. Duan et al. (2017) and Guo et al. (2018) realized that reusing intermediate parameters during the photogrammetry processing can be a solution. In the SfM steps of photogrammetry, the relative camera positions and rotation angles between the raw images and the field are estimated. Consequently, the transformation matrix that maps raw image pixel coordinates to actual global geographical coordinates can be derived from the intermediary parameters obtained through photogrammetry.. Although this idea was successfully examined (Duan et al., 2017; Guo et al., 2018), they did not publish user-friendly tools. To our knowledge, no such tools are currently available for this purpose.

Lastly, there are no easy-to-use tools for forward projecting, which converts positions on raw images to those on the 3D model. After object detection or segmentation on raw images, linking these image results back to real-world positions is often necessary. For example, our previous study (Wang et al., 2023) segmented broccoli heads on raw images and returned the head polygons to the real-world positions to link to the corresponding actual crops and calculate their actual sizes. This demand also exists in forest ecology (Weinstein et al., 2022) and animal husbandry (Shao et al., 2020). On the other hand, direct analysis on 3D models or point clouds is often more difficult than image analysis. Especially since point clouds currently lack mature and easy-to-use deep learning analysis tools, mapping the analysis results on images back to 3D can reduce the analysis difficulty. For example, Murata and Noshita (2023) extracted leaf edges on 2D raw images and projected the results to the 3D plant point cloud, which is often difficult for leaf edge extraction. There is a great demand for this intermediate processing, but a lack of handy tools makes this process challenging.

To tackle previously recognized challenges and to reduce the coding burden during the intermediate stages of data preprocessing for breeders and agronomists, we developed the open-source Easy Intermediate Data Processor (EasyIDP (Wang et al., 2021), MIT license) in 2021. We have received ample feedback and suggestions (10 citations and 21 forum responses) as it has become more widely used. The main complaints concern the remaining complexity of the API and the lack of detailed documentation. To resolve these issues, we present EasyIDP v2.0, a completely rewritten version that implements detailed documentation, far more simplified APIs, and several new features for easier usage.

2 Methodology

This section introduces the methods and dependencies for the core functional features. The core functional features of EasyIDP v2.0 are shown in Figure 1. Compared to the v1.0 (Wang et al., 2021), which only supported regions of interest (ROI) cropping (Fig. 1-3&4) and backward projection (Fig. 1-1), the v2.0 adds support for forward projection (Fig. 1-2).

2.1 ROI Cropping Module

This module consists of the submodules of ROI loader, photogrammetry outputs loader, and cropped ROI saver.

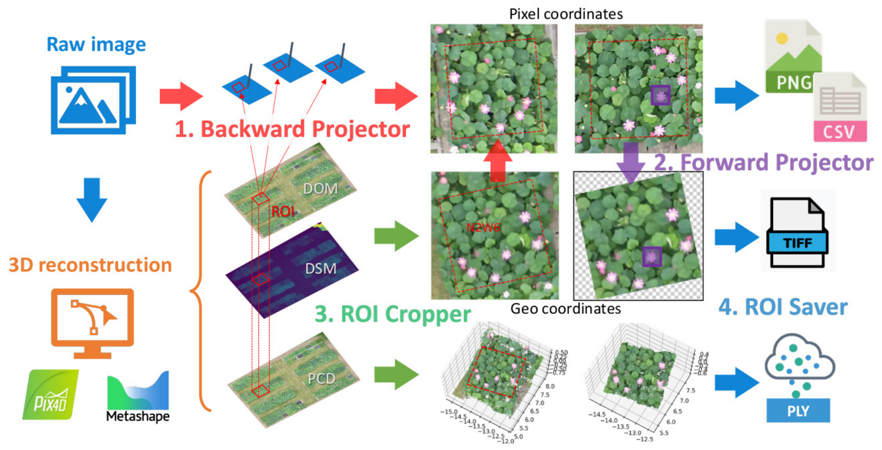


Fig. 1: The core functional modules of EasyIDP v2.0 provide intermediate data processing for photogrammetry-based plant phenotyping. This package is based on the reconstruction projects' intermediate files from either Pix4DMapper (Pix4D SA, Switzerland; `https://pix4d.com`) or Metashape (Agisoft LLC, St. Petersburg, Russia; `https://www.agisoft.com`)

2.1.1 Creating ROI

First, users need to create the ROI files. Since some user-friendly tools already provide a graphical user interface (GUI) for creating ROIs on different formats of photogrammetry outputs, EasyIDP v2.0 does not reinvent the wheel regarding ROI creation. Currently, EasyIDP supports reading ROI files created by LabelMe (`https://github.com/labelmeai/labelme`, free software), QGIS (free software), and CloudCompare (`http://www.cloudcompare.org`, free software), for creating ROIs on raw images (.png, .jpg), geographical maps (.tiff), and 3D point clouds (.ply, *.pcd, *.las) respectively. For more details about creating ROI files for corresponding formats, refer to `https://easyidp.readthedocs.io/en/latest/backgrounds/roi_marking.html` for more details.

2.1.2 Loading ROI

The next step involves loading the ROI files created by different software, including LabelMe, QGIS, and Cloudcompare.

LabelMe produces the *.json format for recording ROI information. The built-in Python "json" library is used to parse the coordinates and labels of ROIs on raw 2D images.

For QGIS or ArcGIS, these produce the *.shp and *.geojson formats for recording ROI information. The external Python packages "pyshp" (`https://github.c` om/GeospatialPython/pyshp) and “geojson” (https://github.com/jazzband/geojson) are used to parse the coordinates and the ROI labels for geographical TIFF images. Cartographic projections and coordinate transformations between different geographical coordinate reference systems (CRS) are supported by “pyproj” (https://github.com/pyproj4/pyproj).

CloudCompare uses the *.txt format to record the information of picked vertices of ROI on 3D point clouds. The external package “numpy” (https://github.com/numpy/numpy, (van der Walt et al., 2011)) is used to load the txt file directly.

2.1.3 Loading Photogrammetry Outputs

The subsequent step is to load the photogrammetry outputs, which commonly include the GeoTiff file of geographical maps (DOM and DSM) and the 3D point cloud of 3D models.

For loading the GeoTiff files, the external package “tifffile” (https://github.com/cgohlke/tifffile) is used to load the metadata (header) and image pixel data. The metadata records the resolution, offsets of the top-left corner, and geographic projection information, which is crucial to transform pixel coordinates into real-world geographical coordinates. To handle large-size GeoTiff on computers with low performance, the image pixel data is not fully loaded at once into the computer’s memory. The loader module only saves the metadata; only the required regions are partially loaded to access the image data. This operational logic sacrifices some computation performance but enables the handling of GeoTiff files larger than the computer’s memory.

For loading the point cloud files, unlike v1.0, which used the external package “open3d” (http://www.open3d.org, (Zhou et al., 2018)), v2.0 replaced this heavy external package (> 50MB) with lighter ones (< 2MB), including “plyfile” (https://github.com/dranjan/python-plyfile) and “laspy” (https://github.com/laspy/laspy) for handling *.ply and *.las files, respectively. In contrast to image arrays that store pixel data in a structured manner from left to right and top to bottom, point clouds are often saved in a disorganized fashion. Therefore,, the partial loading strategy is unsuitable;. Limited by this, EasyIDP v2.0 completely loads all the data from point cloud files. As points are sparsely distributed, point clouds typically have much smaller file sizes and memory requirements compared to GeoTiff files, enabling the handling of 100 million points with just 2GB of memory.

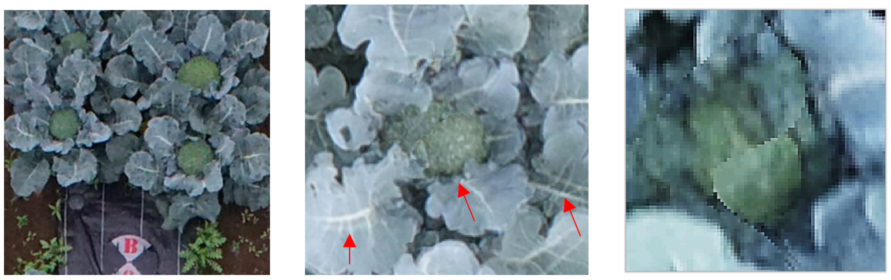
2.1.4 ROI Cropping and Saving

For GeoTiff cropping, the bounding box for each ROI was firstly calculated, then use the “tifffile” package to partially load the rows and cut the columns within the bounding box. Subsequently, the binary mask of the ROI was generated. Suppose the GeoTiff has only a single channel (such as a height map or DSM). In that case, the area outside with nodata values found in the GeoTiff’s metadata is filled, as is common in GeoTiff definitions. Suppose the GeoTiff contains three RGB channels (such as an orthomosaic map or DOM). In that case,the binary mask as the fourth (alpha) channel is added, following the same definition as PNG transparency. Suppose the GeoTiff has more than three channels, indicating a multi-spectral map. In that case, each channel in the same manner is processed as the single-channel case described earlier. Lastly, the offset from the top-left corner of the ROI to the original top-left corner of the GeoTiff is calculated and is updated this metadata in the cropped results. These results are also saved using the “tifffile” package.

For point cloud cropping, the bounding box of each ROI within the point cloud is calculated. Then, points within that bounding box can be rapidly filtered by simply comparing their x and y axis values. The `contains_points()` function from the “matplotlib” package is used to filter points inside the ROI for the remaining points in the bounding box. The results are saved to *.ply or *.las files using the “plyfile” or “laspy” package.

2.2 Backward Projection Module

This backward projection aims to project ROIs from world coordinates (GeoTiffs/DOM with geo-references) to their corresponding raw images (pixel coordinate without geo-references). The benefits of doing such an operation include that the raw image often has better image quality for accurate analysis (Fig. 2a) than DOM. Since the SfM algorithms assume that the object for reconstruction is solid and rigid, its shape should not change from different image views. However, the in-field condition is complex. The plant canopy or leaf is easily moved by wind, which causes its shape to change in different images. This condition conflicts with the algorithm assumption and often causes ”ghost effects” (Fig. 2b) and ”seamline distortion” (Fig. 2c).



(a) raw image (b) DOM: ghost effects (c) DOM: seamline distortion

Fig. 2: The comparison between orthomasic and raw image

This module uses, the built-in Python packages “zipfile” and “xml” are used to import both the external and internal camera parameters produced by photogrammetry projects. Then, the backward calculation is applied using the pinhole camera model. Finally, camera distortion calibration is performed.

2.2.1 Camera Parameters Loading

The geometry connection between the raw images and field is determined upon executing the photogrammetry software. It is consisted of two principal components: the external and internal parameters. The external ones are unique to each raw image, encompass the camera's (x, y, z) location in real-world coordinates. (O_{world}, Fig. 3a) and the camera's (yaw, pitch, roll) rotation. The internal ones detail the attributes of the sensor and are constant for each raw image. They cover the focal length, size of the camera's charge-coupled device (CCD), and the calibration parameters for lens distortion. For more information, see `https://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined`. These parameters are available under the intermediate files of the Agisoft Metashape (Agisoft LLC, St. Petersburg, Russia; `https://www.agisoft.com`) and Pix4D (Pix4D SA, Switzerland; `https://pix4d.com`) project intermediate files.

In Pix4D projects, can find all these parameters within the "1_initial/params" folder. Files such as "calibrated internal camera parameters.cam," "calibrated camera parameters.txt," "pmatrix.txt," and "offset.xyz" are imported directly as text files and interpreted within the EasyIDP package, eliminating the need for any supplementary packages. For additional information regarding these files, please consult the official documentation provided by Pix4D (`https://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined`).

In Agisoft Metashape projects, these parameters can be accessed by utilizing APIs (which requires a Professional license) or by extracting them from the zipped project xml file "project.files/0/chunks.zip/doc.xml". The EasyIDP package selects the latter approach, thereby does not need for a professional license. The "zipfile" and "xml" packages are utilized to decompress and interpret the parameters within the XML files.

2.2.2 Backward Projection Formulas

The transformation of geometry from the real-world coordinate (O_{world}) to the image pixel coordinate (O_{pix}) is shown in Fig. 3a-c. Four distinct coordinate systems exist, with O_{world} being the first, typically measured in meters (Fig. 3a). The camera coordinate system (O_{cam}, Fig. 3b) represents the second system, positioning the camera at the coordinate origin, denoted as $(0, 0, 0)$. The optical axis of the camera is in the line with the z-axis, and typically,, the point O_{img} does not the same with the central point of the plane. The third system is the camera CCD coordinate (O_{img}, Fig. 3c), with millimeters being the standard unit of measurement. The last system

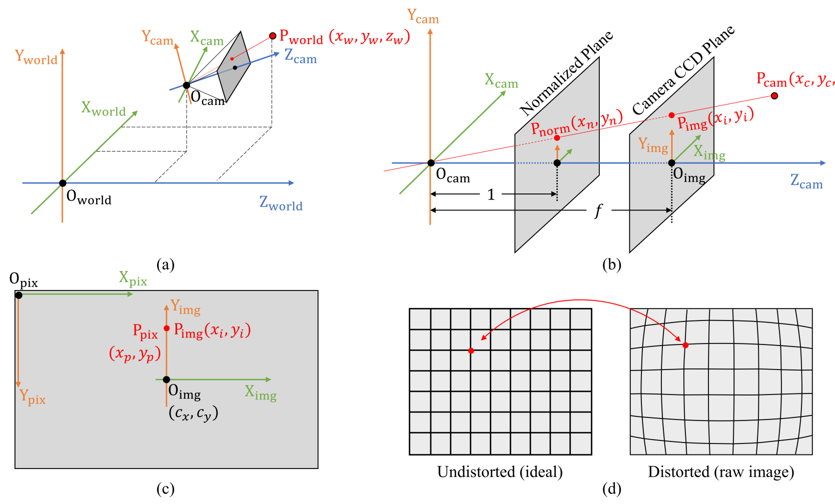


Fig. 3: Projecting a point backward from the 3D world coordiantes to the 2D pixel coordinates present on raw UAV images, utlizing the pinhole camera model (Wang et al. (2021), Fig. 2). (a) The relationship between the world coordinate system (O_{world}) and the camera coordinate system (O_{cam}) is established through the camera's external parameres, which include its position and rotation. (b) The relationship between the camera coordinate system (O_{cam}) and the image coordinate system (O_{img}). (c) The relationship between the image coordinate system (O_{img}) and the pixel coordinate system (O_{pix}). (d) Calibrating the lens-induced distortion from distorted to undistorted images.

is the pixel coordinate (O_{pix}), originating from the upper-left corner of O_{img}, and the scale is in pixels.

Assuming a point $P_{world}(x_w, y_w, z_w)$ in O_{world}, to transform that point to $P_{cam}(x_c, y_c, z_c)$ in O_{cam} (Fig. 3a), a transformation matrix T measuring 3×4 can be obtained using the camera's position (t, representing the translational transformation) along side the camera's rotation (R, indicating the rotational transformation):

$$\begin{aligned} P_{cam} &= T \cdot P_{world} \\ \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} &= \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{11} & R_{12} & R_{13} & t_1 \\ R_{11} & R_{12} & R_{13} & t_1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \end{aligned} \tag{1}$$

Where t is a 3×1 matrix representing camera position, and R is the 3×3 rotation matrix calculated from camera rotation parameters, which include yaw, pitch, and roll, denoted by $(\omega, \varphi, \kappa)$ (`https://support.pix4d.com/hc/en-us/articles/202558969-Yaw-Pitch-Roll-and-Omega-Phi-Kappa-angles`):

$$
\begin{aligned}
R &= R(\omega)R(\varphi)R(\kappa) \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\omega) & -sin(\omega) \\ 0 & sin(\omega) & cos(\omega) \end{bmatrix} \begin{bmatrix} cos(\varphi) & 0 & sin(\varphi) \\ 0 & 1 & 0 \\ -sin(\varphi) & 0 & cos(\varphi) \end{bmatrix} \begin{bmatrix} cos(\kappa) & -sin(\kappa) & 0 \\ sin(\kappa) & cos(\kappa) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cos\kappa cos\varphi & -sin\kappa cos\varphi & sin\varphi \\ cos\kappa sin\omega sin\varphi + sin\kappa cos\omega & cos\kappa cos\omega - sin\kappa sin\omega sin\varphi & -sin\omega cos\varphi \\ sin\kappa sin\omega - cos\kappa cos\omega sin\varphi & sin\kappa cos\omega sin\varphi + cos\kappa sin\omega & cos\omega cos\varphi \end{bmatrix} \\
&= \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{11} & R_{12} & R_{13} \\ R_{11} & R_{12} & R_{13} \end{bmatrix}
\end{aligned}
\tag{2}
$$

The distance between the normalized plane and the origin O_{cam} measures 1 mm, whereas the gap from the camera's CCD plane to the origin O_{cam} corresponds to the focal length f as shwon in mm inFig. 3b. The transformation from $P_{cam}(x_c, y_c, z_c)$ to the normalized plane $P_{norm}(x_n, y_n)$ and to the camera CCD plane $P_{img}(x_i, y_i)$ can be derived by triangle similarity:

$$
\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = f \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = f \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{3}
$$

The conversion of $P_{img}(x_i, y_i)$ expressed in millimeters to the pixel coordinate position $P_{pix}(x_p, y_p)$ (Fig. 3c), requires the application of this equation series:

$$
\begin{cases} x_p = \alpha \cdot x_i + c_x = f \cdot \alpha \cdot x_n + c_x = f_\alpha \cdot x_n + c_x \\ y_p = \beta \cdot y_i + c_y = f \cdot \beta \cdot y_n + c_y = f_\beta \cdot y_x + c_y \end{cases} \tag{4}
$$

where α and β represent the pixel resolution in pixels per millimeter, which typically matches in models of pinhole cameras. The focal length in pixels is represented by f_α and f_β. It is important to note that with Pix4D, (c_x, c_y) can be obtained directly, while for Agisoft Metashape (`https://www.agisoft.com/pdf/metashape-pro_1_7_en.pdf`, p. 176). In the XML file, (c_x, c_y) does not correspond to the definition provided here; it is the deviation from the image's central point, which is equivalent to $(0.5w + c_x, 0.5h + c_y)$, with w and h representing the width and height in pixels, respectively.

Equations (4) can be expressed in the following homogeneous coordinate form:

$$
\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_\alpha & 0 & c_x \\ 0 & f_\beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = K \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \tag{5}
$$

By summing up Equations (1) to (5), $P_w(x_w, y_w, z_w)$ can be directly transformed to $P_{pix}(x_p, y_p)$, which can be calculated by:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = K \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \frac{1}{z_c} K \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} K & 1 \end{bmatrix} T \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = P_{mat} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{6}$$

the matrix P_{mat}, which is a 3×4 array, is frequently referred to as the projection matrix. It is capable of directly converting points from three-dimensional global coordinates to two-dimensional pixel coordinates.

2.2.3 Camera Distortion Calibration

Equation (5) represents an idealized transformation, disregarding the distortion introduced by the lens of the camera, as illustrated in Fig. 3.d. To correct this distortion, numerous camera calibration parameters are employed, encompassing three or four coefficients for radial distortion (denoted as K_i in MetaShape and R_i in Pix4D) along with a pair of coefficients for tangential distortion (referred to as P_i in MetaShape and T_i in Pix4D). MetaShape offers coefficients for affinity (B_1) and non-orthogonality (B_2), expressed in pixel units. Adjustments for the distorted pixel coordinates, denoted as (x'_p, y'_p), are specified through the subsequent set of equations:

$$\begin{aligned} \text{Pix4D} &= \begin{cases} x'_p = c_x + x'f \\ y'_p = c_y + y'f \end{cases} \\ \text{Metashape} &= \begin{cases} x'_p = c_x + x'f + x'B_1 + y'B_2 \\ y'_p = c_y + y'f \end{cases} \end{aligned} \tag{7}$$

where:

$$\begin{aligned} \text{Pix4D} &= \begin{cases} x' = k_0 x_n + 2T_2 x_n y_n + T_1(r^2 + 2x_n^2) \\ y' = k_0 y_n + 2T_1 x_n y_n + T_2(r^2 + 2y_n^2) \end{cases} \\ \text{Metashape} &= \begin{cases} x' = k_0 x_n + 2P_2 x_n y_n + P_1(r^2 + 2x_n^2) \\ y' = k_0 y_n + 2P_1 x_n y_n + P_2(r^2 + 2y_n^2) \end{cases} \end{aligned} \tag{8}$$

and:

$$\begin{aligned} r &= \sqrt{x_n^2 + y_n^2} \\ k_0 &= \begin{cases} 1 + R_1 r^2 + R_2 r^4 + R_3 r^6 & \text{(Pix4D)} \\ 1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + K_4 r^8 & \text{(Metashape)} \end{cases} \end{aligned} \tag{9}$$

2.3 Forward Projection Module

By using the backward projection module mentioned above, raw images can be utilized for accureate image analysis tasks such as organ-level detection and segmentation (for example, broccoli head). Due to the raw image containing only pixel coordinates and the real-world scale of each pixel is uncertain and varies. This is applicable for computing relative traits on raw images like counting broccoli numbers (by detection) and calculating canopy coverage (by segmentation and pixel numbers ratio), but presents challenges for obtaining absolute measurements such as length and area. In contrast, orthomosaic images possess a consistent scale across all pixels but often have lower image quality, which is crucial for precise image analysis. Hence, it is essential to transfer the results of image analysis (e.g., segmentation of individual broccoli heads) back to the orthomosaic to determine the accurate length and area, a process known as forward projection.

Nevertheless, performing forward projection poses greater challenges. In the backward projection process, converting 3D geographical coordinates (3D canopy model) to the 2D pixel coordinate (2D raw image), one dimension (z or height) is lost and cannot be reversed. Therefore, when converting the 2D pixel coordinates back to the 3D geographical coordinates, only a ray (Z_{cam}) joining the camera position (O_{cam}) to the point on the camera CCD plane can be produced for each vertex (point) of the broccoli head polygons (Fig. 3a), instead of obtaining exact 3D geographical points.

To obtain accurate 3D positions, there are currently two solutions: 1) intersecting two or more rays from different perspectives to determine the 3D intersection point, similar to the SfM process in photogrammetry and 2) computing the intersection points between the ray and the 3D surface of the broccoli canopy model. For the first solution, matching segmentation polygons from different perspectives for the same broccoli is not difficult, but matching each vertex is challenging. Since the shapes vary slightly in different perspectives, the vertex counts may not align, let alone the vertices with ”random” orders. For the second solution, it is necessary to determine whether the ray intersects with all the triangular faces and calculate the intersection point if it does. This approach requires extensive computation, as a 3D broccoli canopy model typically contains millions of triangular faces, and each broccoli head polygon has dozens of vertices. At the same time, there may be several thousand polygons. Moreover, a ray may intersect with multiple triangular faces, such as leaf and head faces, necessitating the identification of the actual triangular face of the broccoli head. Shao et al. (2020) optimized this approach by projecting the 3D triangular faces onto the raw image to generate a depth image, mitigating the dimension loss. However, this method remains computationally intensive for rendering each pixel, and the code is not publicly available.

Inspired by the depth image rendering proposed by Shao et al. (2020) and the projective transformation used in our previous study (Wang et al., 2023), a control point array covering the ROI and the broccoli segmentation results (Fig. 4a) was generated in geographical coordinates on the Digital Orthophoto Map (DOM). These control points were then backward projected onto the raw images (Fig. 4b), and the piecewise affine transformation was used to revert coordinates according to these control points in both the geographical and image coordinates. This feature is implemented by the “morphology” module of the external package ”scikit-image” (`https://scikit-image.org`, (van der Walt et al., 2014)).

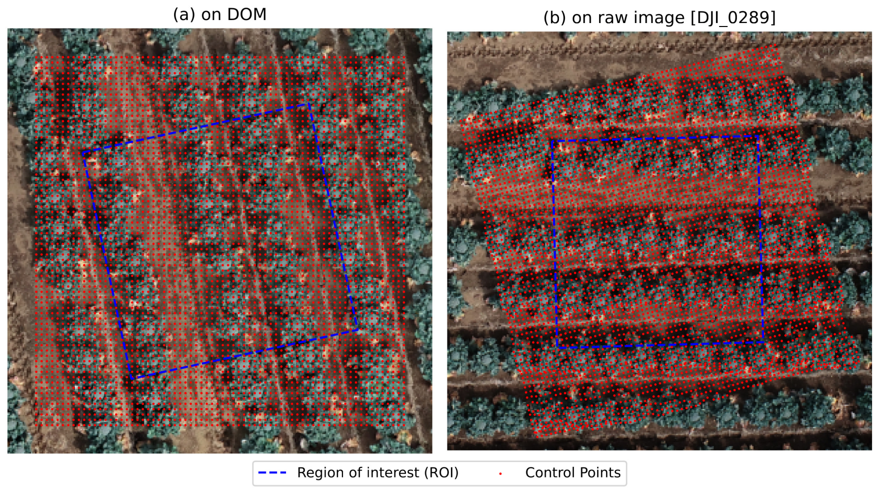


Fig. 4: Control points between geographical coordinates on DOM and the pixel coordinates on a raw image. The blue broken lines represent the ROI boundary of the current grid, and the red dots are control points.

2.4 Implementation

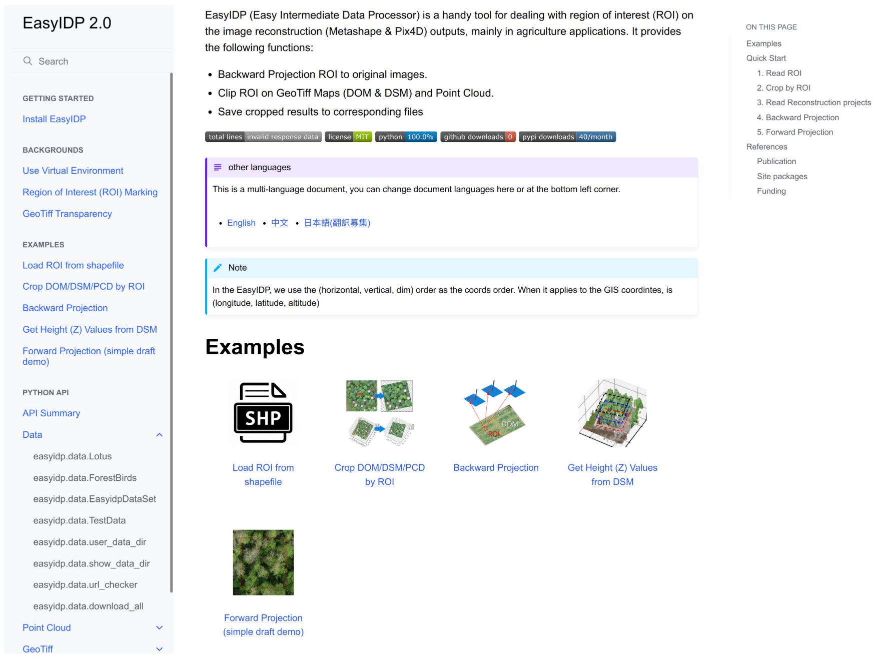
The aforementioned core functional features and other user-friendly features, including PyPI installation and access to example data, have been implemented into EasyIDP v2.0 in Python 3.9. EasyIDP v2.0 is a cross-platform program that functions on Microsoft Windows, MacOS, and GNU/Linux under the MIT license, due to the characteristics of the Python language. The source code is available at `https://github.com/UTokyo-FieldPhenomics-Lab/EasyIDP`. The multi-language package documentation can be found at `https://easyidp.readthedocs.io`. For common scenarios in high-throughput phenotyping analysis, more than 8GB of RAM and a CPU with 3.0GHz or higher are recommended. Nonetheless, it is noteworthy that this tool can still function on less powerful computers, which will will be explored extensively in the discussion section.

3 New Features

This section presents some new features that were implemented in v2.0 compared to the previous version. By adding multilingual documentation, PyPI installation, dataset assessment APIs, and simplifying the core feature APIs, v2.0 significantly decreases the amount of code required for use.

3.1 Add Detailed Documentation

Detailed documentation for EasyIDP v2.0 has been created using reStructuredText (RST). This documentation is also open-source and is hosted on the Read the Docs (`https://easyidp.readthedocs.io`) website (Fig. 5), a renowned documentation hosting platform. The documentation is multilingual; the base language is English, but partial Chinese and Japanese translations are provided. In addition to providing detailed API documentation, instructions for installation and relevant background knowledge are also given. Five executable examples that cover the core features of EasyIDP and intermediate processing for plant phenotyping are also provided.


EasyIDP 2.0

Search

GETTING STARTED

Install EasyIDP

BACKGROUNDS

Use Virtual Environment

Region of Interest (ROI) Marking

GeoTiff Transparency

EXAMPLES

Load ROI from shapefile

Crop DOM/DSM/PCD by ROI

Backward Projection

Get Height (Z) Values from DSM

Forward Projection (simple draft demo)

PYTHON API

API Summary

Data

easyidp.data.Lotus

easyidp.data.ForestBirds

easyidp.data.EasyidpDataSet

easyidp.data.TestData

easyidp.data.user_data_dir

easyidp.data.show_data_dir

easyidp.data.url_checker

easyidp.data.download_all

Point Cloud

GeoTiff

EasyIDP (Easy Intermediate Data Processor) is a handy tool for dealing with region of interest (ROI) on the image reconstruction (Metashape & Pix4D) outputs, mainly in agriculture applications. It provides the following functions:

- Backward Projection ROI to original images.
- Clip ROI on GeoTiff Maps (DOM & DSM) and Point Cloud.
- Save cropped results to corresponding files

total lines | invalid response data   license | MIT   python | 100.0%   github downloads | 0   pypi downloads | 40/month

other languages

This is a multi-language document, you can change document languages here or at the bottom left corner.

- English • 中文 • 日本語(翻訳募集)

Note

In the EasyIDP, we use the (horizontal, vertical, dim) order as the coords order. When it applies to the GIS coordintes, is (longitude, latitude, altitude)

Examples

Load ROI from shapefile

Crop DOM/DSM/PCD by ROI

Backward Projection

Get Height (Z) Values from DSM

Forward Projection (simple draft demo)

ON THIS PAGE

Examples

Quick Start

1. Read ROI
2. Crop by ROI
3. Read Reconstruction projects
4. Backward Projection
5. Forward Projection

References

Publication

Site packages

Funding

Fig. 5: A screenshot of the documentation homepage.

3.2 Add PyPI Installation Support

EasyIDP could only be used in the previous version by downloading the source code. In version 2.0, PyPI installation is supported. It can be installed with a single bash command: `pip install easyidp`.

To verify successful installation, please using the following Python script:

Python Code

```
>>> import easyidp as idp
>>> idp.__version__
2.0.0
```

For more detailed installation instructions on different operating systems, please refer to the documentation: `https://easyidp.readthedocs.io/en/latest/install.html`

3.3 Add Example Dataset API

Providing beginners with example data to learn how to use the package is necessary, but sharing this data has been problematic. Previously, the example data was uploaded to Google Drive and the download link was provided for manually downloading. Many users complained that after downloading the example data, it was still difficult to run the demo code due to varying download paths, which often needed to be changed in different places. Therefore, version 2.0 implements the example dataset API to handle the downloading and path management steps.

Currently, EasyIDP v2.0 provides two datasets as examples: the Lotus plot in Tanashi, Tokyo, Japan, from 2017 (Fig. 6a) and the Forest Bird survey in Florida, US, from 2022 (Fig. 6b). The datasets include parts of raw images, photogrammetry projects by Pix4D and Metashape, plot boundary ROIs, and output maps. It is shown below, how the dataset API is used:

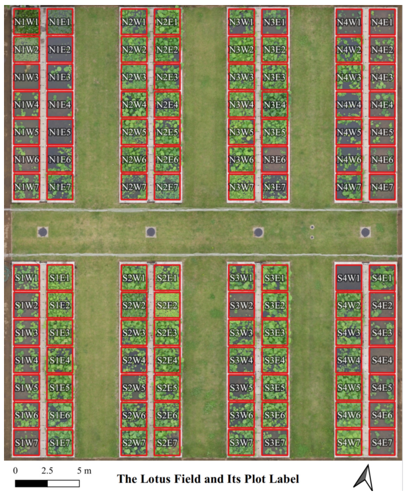
Python Code

First, please ensure that you have imported the easyidp package:

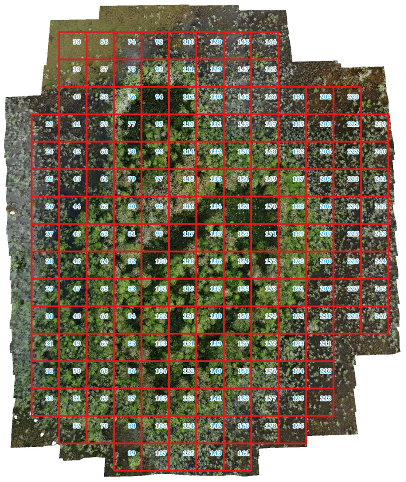
```
>>> import easyidp as idp
```

Then, open the Lotus dataset; it downloads the dataset automatically if it is not available on the local machine:

```
>>> lotus = idp.data.Lotus()
```



(a) The Lotus dataset



(b) The Forest Bird dataset

Fig. 6: The example dataset provided by EasyIDP v2.0

```
Downloading...
From: https://drive.google.com/uc?id=1SJmp-bG5SZrwdeJL-...
To: C:\Users\...\easyidp.data\2017_tanashi_lotus.zip
100% 3.58G/3.58G [00:54<00:00, 65.4MB/s]
```

Here shows how to check and open the downloaded dataset folder:

```
>>> idp.data.user_data_dir()
'/Users/.../Application Support/easyidp.data'
>>> idp.data.show_data_dir()
```

EasyIDP will open the folder by popping up the system's default file explorer. Similarly, the dataset's items can be accessed as follows:

```
>>> lotus.shp
'C:\\Users\\...\\2017_tanashi_lotus\\plots.shp'
>>> lotus.metashape.proj
'C:\\Users\\...\\2017_tanashi_lotus\\170531.Lotus.psx'
>>> lotus.photo
'C:\\Users\\...\\2017_tanashi_lotus\\20170531\\photos'
```

Please refer `https://easyidp.readthedocs.io/en/latest/python_api/data.html` for more details about the dataset API.

3.4 Change ROI Cropping API

Here shows how the latest ROI Cropping API is used.

Python Code

Open the ROI from a shapefile, with 2D longitude and latitude information:
```
>>> roi = idp.ROI("xxxx.shp")
```

Open the photogrammetry output files, including GeoTiff and point cloud:
```
>>> dom = idp.GeoTiff("xxx_dom.tif")
>>> dsm = idp.GeoTiff("xxx_dsm.tif")
```

Read the point cloud data:
```
>>> ply = idp.PointCloud("xxx_pcd.ply")
```

Crop the ROI:
```
>>> dom_parts = roi.crop(dom)
>>> dsm_parts = roi.crop(dsm)
>>> pcd_parts = roi.crop(ply)
```

Or, crop the ROI and save the results to a given folder:
```
>>> dom_parts = roi.crop(dom, save_folder="./crop_dom")
>>> dsm_parts = roi.crop(dsm, save_folder="./crop_dsm")
>>> pcd_parts = roi.crop(ply, save_folder="./crop_pcd")
```

The `dom_parts`, `dsm_parts`, `pcd_parts` are the cropping ROI results. Please refer `https://easyidp.readthedocs.io/en/latest/jupyter/load_roi.html` for more details about reading ROI.

3.5 Change Backward Projecting API

Here is a simple example of using the backward projection API:

Python Code

Ensure you open an ROI with 3D coordinate:
```
>>> roi = idp.ROI("xxxx.shp")
>>> roi.get_z_from_dsm("xxxx_dsm.tiff")
```

Then open the photogrammetry project:

```
>>> ms = idp.Metashape("xxxx.psx", chunk_id=0)
```

Conduct the backward projection and save the result to the given folder:

```
>>> img_dict = roi.back2raw(ms, save_folder="result/folder/")
```

The `img_dict` is the result of backward projection. For more details about the backward projection, please refer to the example documentation: `https://easyidp.readthedocs.io/en/latest/jupyter/backward_projection.html`.

3.6 Add Forward Projecting API

Similar to the backward projection API, a simple way for the forward projection API is mentioned below:

Python Code

Continue with the previous backward projection:

```
>>> imarray_dict = roi.raw4dom(ms)
```

The `imarray_dict` is the result of forward projection. For more details, please refer to the example documentation: `https://easyidp.readthedocs.io/en/latest/jupyter/forward_projection.html`.

3.7 Add Visualizing API

Lastly, EasyIDP v2.0 also provides an API for visualizing forward projection results.

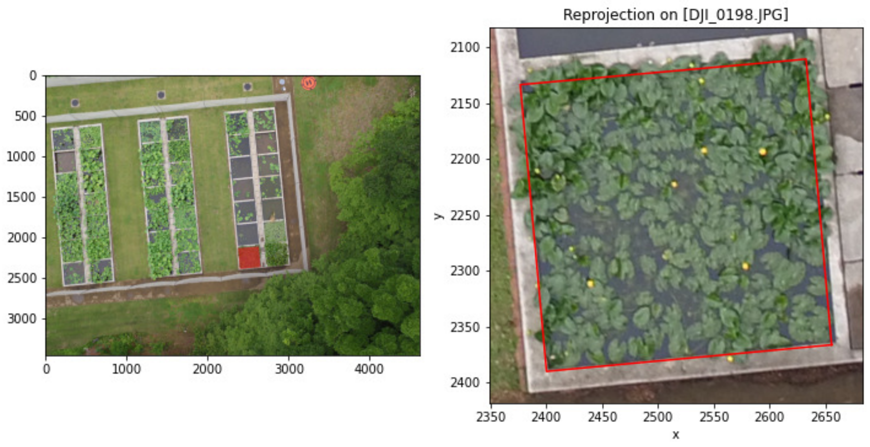
Python Code

The following API draws the backward projection results on one given raw image ‘DJI_0198.JPG’ (Fig. 7a)

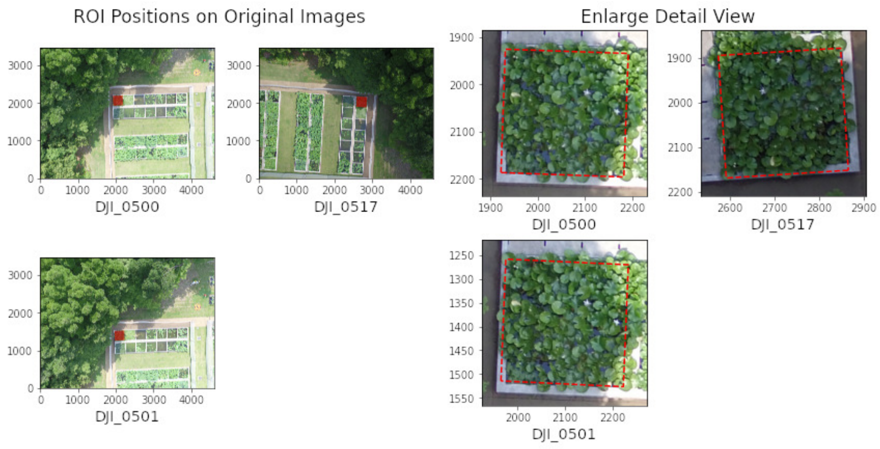
```
>>> img_name = "DJI_0198.JPG"
>>> idp.visualize.draw_polygon_on_img(
...     img_name, photo.path, out_dict[img_name],
...     save_as="p4d_back2raw_single_view.png")
```

The following API draws all backward projection results of one ROI (Fig. 7b):

```
>>> idp.visualize.draw_polygon_on_img(
...     ms, img_dict_ms['N1W1'],
...     save_as="draw_backward_one_roi.png"
... )
```



(a) Visualization of one image by "draw_polygon_on_img()"



(b) Visualization of one ROI on all images "show_roi_on_img()"

Fig. 7: Two visualization functions for backward projection results

For more parameter details about both visualization APIs, please refer to the accompanying documentation: `https://easyidp.readthedocs.io/en/latest/python_api/autodoc/easyidp.visualize.draw_backward_one_roi.html`

4 Discussion

One of the innovations of EasyIDP v2.0 is making intermediate data processing more accessible for agronomists and breeders, even those with only basic Python programming skills.

The EasyIDP package does not necessitate high computational resources, as it does not currently support parallel computing or GPU computing. A computer manufactured within past 10 years with 4GB RAM should suffice for running this package; it even passed testing on a Linux emulator on an Android tablet. However, performing the SfM-MVS reconstruction process is a computationally intensive task that heavily depends on high-performance CPUs and GPUs for high-throughput processing, which is very difficult for real-time. In this scenario, the computational complexity limitations lie more on the photogrammetry SfM-MVS reconstruction side rather than on the EasyIDP side. While EasyIDP does not demand high computational resources, this does not imply that it can efficiently process large files on low-performance computers. Although we attempted to reduce memory usage by cropping large GeoTiff files exceeding memory capacities, this approach sacrifices processing speed compared to loading them entirely into memory. To be more specific, for GeoTiff cropping, since each ROI is processed sequentially, parallel ROIs in the same row lead to redundant read operations, creating a performance bottleneck. For instance, cropping thousands of small ROIs from a $100,000 \times 100,000$ pixel GeoTiff takes over 3 hours to complete. In the future, we aim to enhance this performance by using asynchronous operations for each ROI to reduce the impact of repeated reading and writing. However, it is always recommended to use a computer that meets the computational power requirements to run this software package for an optimal user experience.

Execpt for the previous performance limitation, EasyIDP also has weaknesses in the newly developed forward projection. We assume that the target object resembles an agricultural field – roughly flat with local variance (vegetation canopy), to prevent the control point array from being excessively distorted when back-projected onto the original map. This method is unsuitable for scenarios with drastic morphological changes, such as indoor close-range 3D reconstruction models lacking geographic coordinates. While the Metashape Pro version provides a Python API that could be used for this purpose, it depends on the more expensive Pro version and is not transferable to Pix4D. Further testing and refinement are needed to verify feasibility and identify improvements for some newly supported features in this v2.0 updates. Moreover, our GitHub issue tracker contains over a dozen user-submitted suggestions for enhancement that will be considered for future development.

This kind of open-source and public deployment also contribute to the reproducibility of research, which is currently problematic in plant phenotyping studies and tools. Many studies do not publicly share their datasets or code, making it difficult to reproduce their results. This leads to a waste of resources as similar problems are tackled repeatedly without the benefit of shared experience. As Lobet (2017) stated in the "*Image Analysis in Plant Sciences: Publish Then Perish*" paper, many published phenotyping tools still suffer from severe reproducibility issues and a lack of ease of use. The main issues are: 1) code and example data are often published on personal or laboratory websites, which are slow or inaccessible due to slow download speeds resulting from bandwidth costs, URL changes, and lack of maintenance; 2) only code files and simple introductory files are provided, with detailed instructions for setting up the code environment, runnable examples, and often miss the detailed API documentation, leading to an inability to operate and apply the code to one's data; and 3) a lack of follow-up maintenance and community support, where bugs may be addressed through difficult email communications, but user experiences remain unshared. For EasyIDP v2.0, all source code and documentation are publicly available on renowned platforms such as GitHub and ReadTheDocs, significantly reducing the risk of inaccessibility and the workload for maintenance. We also provide detailed documentation for installation, examples, and API use, enabling users to reproduce the processes with their dataset. Lastly, the GitHub issue and discussion features are utilized as tools for maintenance and community support to respond to questions and collect new feature requests. All these abovementioned points help to establish EasyIDP v2.0 as a reliable, long-term support project for researchers in the plant phenotyping field.

5 Conclusions

The EasyIDP package was proposed to reduce the workload of intermediate data processing in plant phenotyping. This v2.0 update completely rebuilds the package and enhances the user experience. It became easier to install, the example datasets were made more accessible, detailed documentation was provided, and the package APIs were streamlined and simplified. Despite the fact that the v2.0 version is still a work in progress with issues such as GeoTiff cropping performance and the robustness of forward projection, the package demonstrates significant potential to contribute to the research community with its open source and public deployment. The improvements in version 2.0 are not limited to the field of phenomics. Applications that integrate SfM-MVS reconstruction outputs and raw images can benefit from these tools. Some of our users have reported using them for forestry inventory, ecological surveys, and urban greening monitoring. EasyIDP is expected to have a bright future in leading to better data analysis experiences for more fields.

Acknowledgements We thank the Institute for Sustainable Agro-ecosystem Services (ISAS) technical staff for managing the lotus field and data collection, as shown in Figure 6a. We also thank

Prof. Benjamin Weinstein from the University of Florida for sharing their dataset of avian ecology surveys, also referenced in Figure 6b. Furthermore, we are grateful to Kunihiro Kodama in our lab for the insightful discussions about MetaShape backward projection.

Competing Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethics Approval No ethics approval was required for this study.

References

Chen CJ, Zhang Z (2020) GRID: A Python Package for Field Plot Phenotyping Using Aerial Images. Remote Sensing 12(11):1697, doi:10.3390/rs12111697

Duan T, Zheng B, Guo W, Ninomiya S, Guo Y, Chapman SC (2017) 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 44(1):169, doi:10.1071/FP16123

Gao T, Zhu F, Paul P, Sandhu J, Doku HA, Sun J, Pan Y, Staswick P, Walia H, Yu H (2021) Novel 3D Imaging Systems for High-Throughput Phenotyping of Plants. Remote Sensing 13(11):2113, doi:10.3390/rs13112113

Gerland P, Raftery AE, Ševčíková H, Li N, Gu D, Spoorenberg T, Alkema L, Fosdick BK, Chunn J, Lalic N, Bay G, Buettner T, Heilig GK, Wilmoth J (2014) World population stabilization unlikely this century. Science 346(6206):234–237, doi:10.1126/science.1257469

Guo A, Huang W, Dong Y, Ye H, Ma H, Liu B, Wu W, Ren Y, Ruan C, Geng Y (2021a) Wheat Yellow Rust Detection Using UAV-Based Hyperspectral Technology. Remote Sensing 13(1):123, doi:10.3390/rs13010123

Guo W, Zheng B, Potgieter AB, Diot J, Watanabe K, Noshita K, Jordan DR, Wang X, Watson J, Ninomiya S, Chapman SC (2018) Aerial Imagery Analysis – Quantifying Appearance and Number of Sorghum Heads for Applications in Breeding and Agronomy. Frontiers in Plant Science 9:1544, doi:10.3389/fpls.2018.01544

Guo W, Carroll ME, Singh A, Swetnam TL, Merchant N, Sarkar S, Singh AK, Ganapathysubramanian B (2021b) UAS-Based Plant Phenotyping for Research and Breeding Applications. Plant Phenomics 2021:1–21, doi:10.34133/2021/9840192

Herrero-Huerta M, Gonzalez-Aguilera D, Yang Y (2023) Structural Component Phenotypic Traits from Individual Maize Skeletonization by UAS-Based Structure-from-Motion Photogrammetry. Drones 7(2):108, doi:10.3390/drones7020108

Hu P, Guo W, Chapman SC, Guo Y, Zheng B (2019) Pixel size of aerial imagery constrains the applications of unmanned aerial vehicle in crop breeding. ISPRS Journal of Photogrammetry and Remote Sensing 154:1–9, doi:10.1016/j.isprsjprs.2019.05.008

Hu P, Chapman SC, Zheng B (2021) Coupling of machine learning methods to improve estimation of ground coverage from unmanned aerial vehicle (UAV) imagery for high-throughput phenotyping of crops. Functional Plant Biology doi:10.1071/FP20309

Lin YC, Zhou T, Wang T, Crawford M, Habib A (2021) New orthophoto generation strategies from UAV and ground remote sensing platforms for high-throughput phenotyping. Remote Sensing 13(5):860, doi:10.3390/rs13050860

Liu F, Song Q, Zhao J, Mao L, Bu H, Hu Y, Zhu XG (2021) Canopy occupation volume as an indicator of canopy photosynthetic capacity. New Phytologist 232(2):941–956, doi:10.1111/nph.17611

Lobet G (2017) Image Analysis in Plant Sciences: Publish Then Perish. Trends in Plant Science 22(7):559–566, doi:10.1016/j.tplants.2017.05.002

Lu N, Wu Y, Zheng H, Yao X, Zhu Y, Cao W, Cheng T (2022) An assessment of multi-view spectral information from UAV-based color-infrared images for improved estimation of nitrogen nutrition status in winter wheat. Precision Agriculture 23:1653–1674, doi:10.1007/s11119-022-09901-7

Mortensen A, Laursen M, Jørgensen R, Gislum R (2019) Drone dataflow - a MATLAB toolbox for extracting plots from images captured by a UAV. In: Precision agriculture '19, Wageningen Academic Publishers, Montpellier, France, pp 959–965, doi:10.3920/978-90-8686-888-9_118

Murata H, Noshita K (2023) Three-dimensional leaf edge reconstruction using a combination of two- and three-dimensional phenotyping approaches. doi:10.21203/rs.3.rs-3347414/v1

Oats R, Escobar-Wolf R, Oommen T (2019) Evaluation of Photogrammetry and Inclusion of Control Points: Significance for Infrastructure Monitoring. Data 4(1):42, doi:10.3390/data4010042

Okura F (2022) 3D modeling and reconstruction of plants and trees: A cross-cutting review across computer graphics, vision, and plant phenotyping. Breeding Science 72(1):31–47, doi:10.1270/jsbbs.21074

Paulus S (2019) Measuring crops in 3D: Using geometry for plant phenotyping. Plant Methods 15(1):103, doi:10.1186/s13007-019-0490-0

Pivoto D, Waquil PD, Talamini E, Finocchio CPS, Dalla Corte VF, de Vargas Mores G (2018) Scientific development of smart farming technologies and their application in Brazil. Information Processing in Agriculture 5(1):21–32, doi:10.1016/j.inpa.2017.12.002

Ray DK, Mueller ND, West PC, Foley JA (2013) Yield Trends Are Insufficient to Double Global Crop Production by 2050. PLoS ONE 8(6):e66428, doi:10.1371/journal.pone.0066428

Shao W, Kawakami R, Yoshihashi R, You S, Kawase H, Naemura T (2020) Cattle detection and counting in UAV images based on convolutional neural networks. International Journal of Remote Sensing 41(1):31–52, doi:10.1080/01431161.2019.1624858

Su W, Zhu D, Huang J, Guo H (2018) Estimation of the vertical leaf area profile of corn (Zea mays) plants using terrestrial laser scanning (TLS). Computers and Electronics in Agriculture 150:5–13, doi:10.1016/j.compag.2018.03.037

Tresch L, Mu Y, Itoh A, Kaga A, Taguchi K, Hirafuji M, Ninomiya S, Guo W (2019) Easy MPE: Extraction of Quality Microplot Images for UAV-Based High-Throughput Field Phenotyping. Plant Phenomics 2019:1–9, doi:10.34133/2019/2591849

Velumani K, Lopez-Lozano R, Madec S, Guo W, Gillet J, Comar A, Baret F (2021) Estimates of maize plant density from UAV RGB images using Faster-RCNN detection model: Impact of the spatial resolution. Plant Phenomics 2021:9824843, doi:10.34133/2021/9824843

van der Walt S, Colbert SC, Varoquaux G (2011) The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science Engineering 13(2):22–30, doi:10.1109/MCSE.2011.37

van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T (2014) Scikit-image: Image processing in Python. PeerJ 2:e453, doi:10.7717/peerj.453

Wang H, Duan Y, Shi Y, Kato Y, Ninomiya S, Guo W (2021) EasyIDP: A python package for intermediate data processing in UAV-based plant phenotyping. Remote Sensing 13(13):2622, doi:10.3390/rs13132622

Wang H, Li T, Nishida E, Kato Y, Fukano Y, Guo W (2023) Drone-Based Harvest Data Prediction Can Reduce On-Farm Food Loss and Improve Farmer Income. Plant Phenomics 5:0086, doi:10.34133/plantphenomics.0086

Weinstein BG, Garner L, Saccomanno VR, Steinkraus A, Ortega A, Brush K, Yenni G, McKellar AE, Converse R, Lippitt CD, Wegmann A, Holmes ND, Edney AJ, Hart T, Jessopp MJ, Clarke RH, Marchowski D, Senyondo H, Dotson R, White EP, Frederick P, Ernest SKM (2022) A general deep learning model for bird detection in high-resolution airborne imagery. Ecological Applications 32(8):e2694, doi:10.1002/eap.2694

Yang B, Xu Y (2021) Applications of deep-learning approaches in horticultural research: A review. Horticulture Research 8(1):1–31, doi:10.1038/s41438-021-00560-9

Zermas D, Morellas V, Mulla D, Papanikolopoulos N (2020) 3D model processing for high throughput phenotype extraction – the case of corn. Computers and Electronics in Agriculture 172:105047, doi:10.1016/j.compag.2019.105047

Zhou QY, Park J, Koltun V (2018) Open3D: A Modern Library for 3D Data Processing. arXiv 1801.09847