Article

Automating Ground Control Point Detection in Drone Imagery: From Computer Vision to Deep Learning

Gonzalo Muradás Odriozola [1,2], Klaas Pauly [3,*], Samuel Oswald [3] and Dries Raymaekers [3]

[1] Image and Speech Processing (PSI), Department of Electrical Engineering (ESAT), KU Leuven, B-3000 Leuven, Belgium; gonzalo.muradasodriozola@kuleuven.be
[2] Laboratory for Experimental Psychology, Faculty of Psychology and Educational Sciences, KU Leuven, B-3000 Leuven, Belgium
[3] Remote Sensing Unit, Flemish Institute for Technological Research (VITO), Boeretang 200, B-2400 Mol, Belgium; sam.oswald@vito.be (S.O.); dries.raymaekers@vito.be (D.R.)
* Correspondence: klaas.pauly@vito.be

Abstract: Drone-based photogrammetry typically requires the task of georeferencing aerial images by detecting the center of Ground Points (GCPs) placed in the field. Since this is a very labor-intensive task, it could benefit greatly from automation. In this study, we explore the extent to which traditional computer vision approaches can be generalized to deal with variability in real-world drone data sets and focus on training different residual neural networks (ResNet) to improve generalization. The models were trained to detect single keypoints of fixed-sized image tiles with a historic collection of drone-based Red–Green–Blue (RGB) images with black and white GCP markers in which the center was manually labeled by experienced photogrammetry operators. Different depths of ResNets and various hyperparameters (learning rate, batch size) were tested. The best results reached sub-pixel accuracy with a mean absolute error of 0.586. The paper demonstrates that this approach to drone-based mapping is a promising and effective way to reduce the human workload required for georeferencing aerial images.

Keywords: drones; photogrammetry; ground control points; GCPs; RGB; computer vision; deep learning; ResNet; CNN

check for updates

Citation: Muradás Odriozola, G.; Pauly, K.; Samuel, O.; Raymaekers, D. Automating Ground Control Point Detection in Drone Imagery: From Computer Vision to Deep Learning. *Remote Sens.* **2024**, *16*, 794. https://doi.org/10.3390/rs16050794

Academic Editors: Lidia M. Ortega Alvarado and María I. Ramos Galan

Received: 24 November 2023
Revised: 14 February 2024
Accepted: 19 February 2024
Published: 24 February 2024

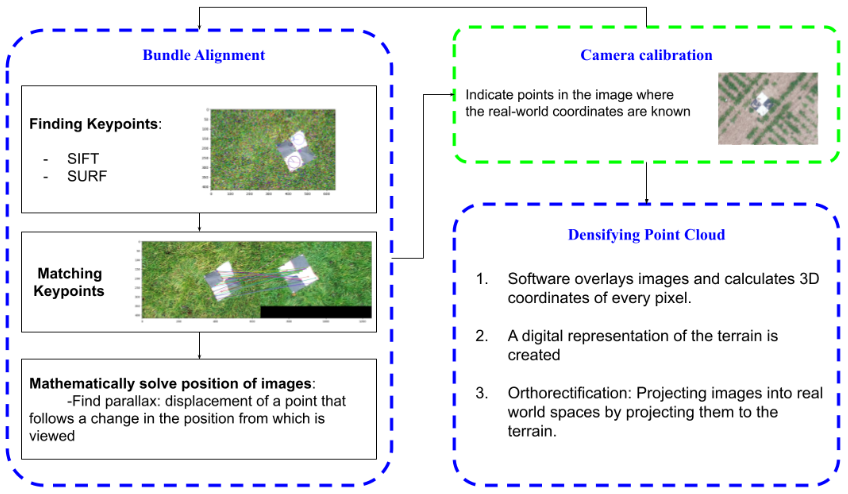
1. Introduction

The utilization of camera-equipped Uncrewed Airborne Systems (UASs), commonly referred to as drones, within the realm of geomatics has experienced a notable surge in recent years. This can be attributed to advancements in sensor quality, cost reduction, and enhanced integration, all of which have significantly improved accessibility and usability [1]. The raw data procured by UAS are typically used in a a meticulous photogrammetric process to generate metric, survey-grade 3D deliverables. Equipped with Global Navigation Satellite Systems (GNSSs), these UASs capture center coordinates for each individual image acquisition point, with contemporary systems offering Real-Time Kinematic (RTK) correction, thereby registering image positions at the centimeter level and providing better constraints to the photogrammetric algorithms.

The photogrammetric process then consists of bundle adjustment, camera self-calibration, dense point cloud generation, orthorectification, and mosaicing of the images. These steps collectively rectify errors present in the initially recorded sensor positions and interior and exterior sensor orientation values [2,3].

Figure 1 shows the photogrammetric process. Starting with bundle adjustment or alignment, this step establishes connections between overlapping images by identifying keypoint features within images, using algorithms such as Scale-Invariant Feature Transform (SIFT) [4–7] or Speeded-Up Robust Features (SURFs) [8]. These keypoint features are interlinked across images, culminating in a solution that explains the position and orientation (both interior and exterior) of cameras through tiepoints. Notably, bundle adjustment and georeferencing may be susceptible to substantial errors, necessitating precise tiepoint measurements. This is accommodated within the camera calibration step, where the bundle adjustment process is reiterated based on the specification of pixel coordinates of Ground Control Points (GCPs) of known real-world coordinates.

Subsequently, the densification of the point cloud is executed, whereby a 3D coordinate is determined for every pixel within the image. Following this, a digital representation of the terrain is generated onto which orthorectified images are projected and mosaiced.



Bundle Alignment
Finding Keypoints:
- SIFT
- SURF
Matching Keypoints
Mathematically solve position of images:
-Find parallax: displacement of a point that follows a change in the position from which is viewed
Camera calibration
Indicate points in the image where the real-world coordinates are known
Densifying Point Cloud
1. Software overlays images and calculates 3D coordinates of every pixel.
2. A digital representation of the terrain is created
3. Orthorectification: Projecting images into real world spaces by projecting them to the terrain.

Figure 1. Simplified diagram of some steps for the photogrammetry workflow.

The integration of GCPs is an indispensable facet of an accurate photogrammetric process. Even when using RTK or post-processed GNSS-INS (Inertial Navigation System), a limited set of GCPs remains crucial for calibration purposes or for independent accuracy assessments [2]. These markers, comprising easily identifiable shapes that stand out against the background, are strategically positioned on the ground. Their precise world coordinates, acquired using survey-grade instruments such as RTK-GNSS or Post-Processed Kinematics (PPK), are meticulously measured in the field [9]. A proper layout and measurement of GCPs, as well as the precise indication of the point of measurement in the individual images (labeling), has proven crucial in ensuring photogrammetric accuracy [10,11].

Various types of GCP markers exist, predominantly consisting of contrasting surface colors such as black and white arranged in quadrants, while coded targets adhering to recognized standards can be detected by software through computer vision, this method is mainly used in close-range photogrammetry and proves impractical for large-scale aerial imagery. Challenges persist regarding the standardization of algorithms capable of autonomously identifying non-coded GCP marker centers centers in drone-based applications. Consequently, labeling is mostly performed manually to date [12]. This manual process is arduous, time-intensive, prone to human error, and would significantly benefit from accurate automation.

Prior attempts to automate GCP center point detection have employed methodologies like Edge Orientation Histograms [13], complemented by the integration of a Convolutional Neural Network (CNN) to enhance robustness [14]. These efforts aimed to predict the intersection point of 65 cm × 65 cm L-shaped GCPs. However, the accuracy of these approaches was susceptible to confounding features in the imagery, such as high altitudes, overlap with road stripes, or surrounding white dust, impacting their reliability. Other computer-vision-based strategies involved X-marker detection using the Hough transform, achieving an average sub-pixel distance of 0.51 between predictions and centers [15]. However, this method operated on 12 × 12-pixel images extracted by a preliminary algorithm. A purely deep learning-driven approach for keypoint detection might offer better generalization capabilities, accommodating irregularities in data without the prerequisite of an initial algorithm to extract the GCP center image.

Another study devised to streamline the process of GCP detection involved the development of a comprehensive pipeline integrating deep learning techniques to detect a broad position of the marker and then compute the center point with computer vision algorithms such as the Canny edge detection method [16]. It found success in using in situ features such as road signs as substitutes for GCPs; however, such features do not commonly exist in agricultural environments. This research was motivated by the need for TLS and photogrammetry data fusion, where two different types of control point patterns are traditionally used. The model's training utilized a small dataset comprising 246 images with fixed resolutions, showcasing prominently visible markers while exhibiting commendable performance under these conditions; the efficacy of this pipeline remains untested in scenarios involving obscured markers, diverse lighting conditions, and variable resolutions. In practical agricultural applications, GCPs are conventionally installed and retained for successive UAS flights, potentially leading to differences in light conditions, resolutions, and the quality of the markers, with all causing degradation in the control point pattern appearance and necessitating the pipeline's ability to generalize to predict GCPs amid non-ideal circumstances.

Convolutional Neural Networks (CNNs), a subset of Artificial Neural Networks (ANNs), are commonly employed for analyzing visual imagery. Studies have investigated combinations of handcrafted filters like Harris [17] and Hessian [18] detectors with learned CNNs—dubbed Key-Net—to detect keypoints, revealing that increasing CNN complexity does not consistently enhance accuracy. Interestingly, integrating handcrafted filters significantly reduces architectural complexity [19]. When it comes to keypoint matching across different raw images, CNNs outperform handcrafted descriptors in terms of accuracy amid illumination and viewpoint changes [20]. Further comparison between pre-trained CNNs and those trained from scratch suggests pre-trained CNNs fare better with illumination changes, while trained CNNs excel with viewpoint alterations.

Deeper CNNs face a *degradation* issue: as depth increases, accuracy plateaus, not due to overfitting but due to higher training error with additional layers. To tackle this, ResNet employs *skip connections* or *shortcuts* that leap over several layers [21]. Similar to Long Short-Term Memory (LSTM) recurrent neural networks [22], there are two main reasons why it is beneficial to add paths to skip connections in the model. First is the aforementioned accuracy saturation problem, and second, to avoid the issue of vanishing gradients [23], thus resulting in more easily optimizable models. The gating mechanism allows information to go through so-called *information highways* [24]. When comparing ResNet to the performance of other deep CNN models such as VGG16 [25] or Xception network [26], the ResNet model outperformed all others in classification tasks [27]. ResNet has been used in the context of GCP detection for classification purposes. In several recent studies, it has been used in combination with object detection architectures such as retinaNet to accurately extract the bounding boxes of GCP pads [28,29]. Similar approaches have been applied using other object detection architectures, such as object-oriented YOLOv5 [30]. The localization of the center point is performed by extracting the actual center of the bounding boxes or utilizing edge features within them. While these methods can detect GCP pads, including non-standard variants, with very high accuracy, the localization of center points remains an issue in operational agricultural applications. In this context, GCP pads may be partly obscured or degraded, and such methods are likely to be inaccurate.

Originally intended for classification, ResNet's adaptability extends to regression tasks. By adjusting output layers and activation functions, ResNet transforms into a regression model, evident in studies predicting pig weight from 3D images [31]. Specifically in the field of keypoint regression, ResNet has seen a lot of success in recent projects, such as in keypoint regression from a 2D image using image-level supervision [32]. Modifications of models like Masked Loss Residual Convolutional Neural Network (ML-ResNet) have been used for facial keypoint detection with missing labels, producing satisfactory results [33], as well as skeletal-point keypoint detection in single images [34]

This paper aims to explore the resilience of a computer-vision-based approach using real-world data. It proposes a ResNet-based deep learning pipeline specifically to swiftly and accurately detect pixel coordinates of black and white ground control marker center points. The paper suggests utilizing residual networks of various depths and comparing a wide array of training hyperparameters to identify an architecture capable of achieving sub-pixel accuracy. The research will compare both deep learning and computer vision approaches.

2. Materials and Methods

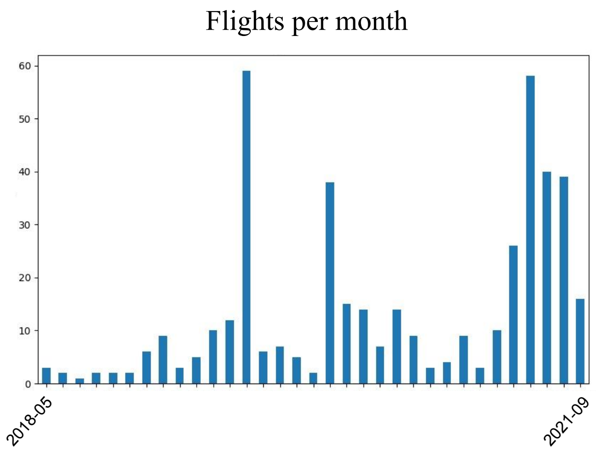
2.1. Existing Data Set

This study started by assembling a curated dataset comprising 2980 JPEG images at their original resolution, gathered between 2018 and 2021. The images were collected in the context of several drone-based agricultural monitoring campaigns across fields in various countries, covering several crop types at different growth stages. All drone flights (474 in total) covered at least 5 GCP markers, and several drone and camera types were used, resulting in image dimensions ranging from 2870 to 5460 pixels in width. Only images containing at least one GCP marker were selected for the current dataset in the framework of this research. For a comprehensive overview, Table 1 details the make and model of all drones utilized in this study, alongside the total number of images captured by each drone during its flights, while Figure 2 shows the geographic and temporal distribution of drone flights. Target ground sample distances ranged from 2 to 10 mm, and the marker sizes within the images varied between 80 and 700 pixels in length. Drones were flown at heights of either 18 or 38 m, depending on target resolution. Figure 3 shows an example of a drone image containing a representative GCP marker in the dataset.

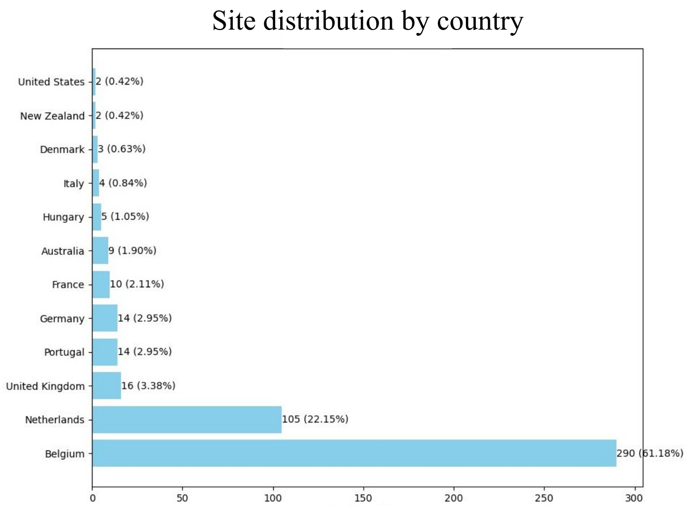
Table 1. Maker, model, and number of pictures taken by each drone used in the study.

Maker Pictures Taken	Model	Pictures Taken
DJI	FC550	120
DJI	FC6310	1077
DJI	FC6310R	763
DJI	FC6310S	25
DJI	FC6510	79
DJI	FC6520	334
DJI	FC6540	132
DJI	M600_X5R	28
DJI	ZenmuseP1	299
Hasselblad	L1D-20c	24
SONY	DSC-RX1RM2	76

The dataset exhibited variations in resolution and flight altitude, leading to diverse sizes of Ground Control Points (GCPs) within the images. All individual images underwent manual annotation by proficient photogrammetry operators who used specialized software to zoom in on the images and click on the GCP markers' center point, thereby assigning a single floating point image coordinate pair to the labeled point. Each operator meticulously assigned a single keypoint, denoted by two floating point image coordinates in pixels, precisely at the center of each GCP.



(a)



(b)

Figure 2. Distribution of sites surveyed (**a**), distribution of flights per month (**b**).

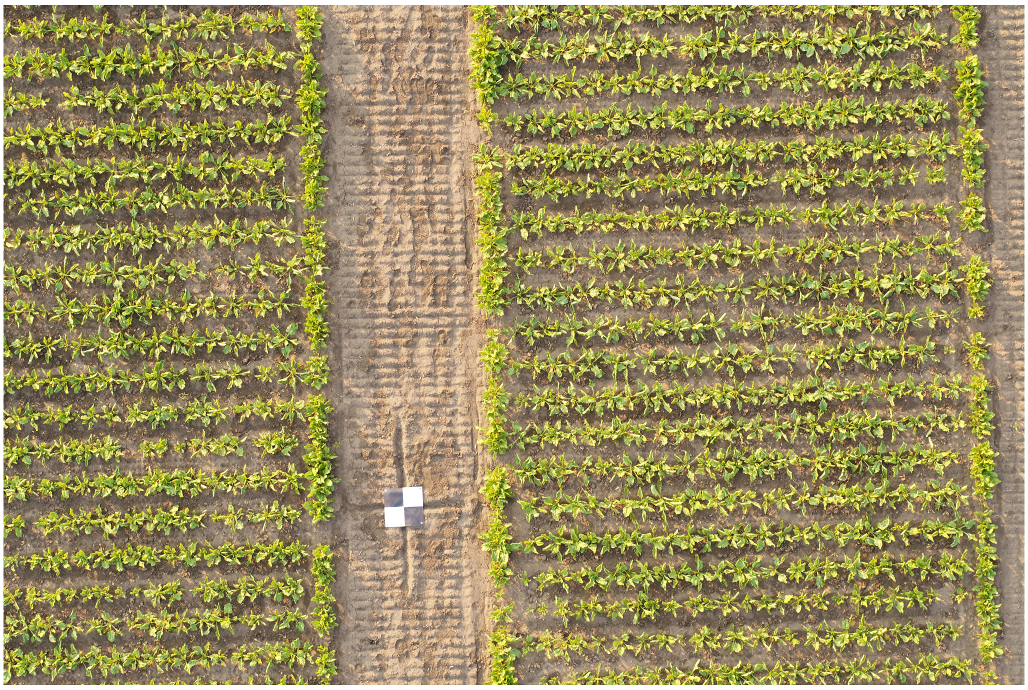


Figure 3. Example of a 6016 × 4008-pixel image captured with a FC6540 with 1/100 s of exposure time, 50 mm of focal length, and 2.97 max aperture.

The dataset includes variability in environmental conditions and GCP marker quality, originating from the different flight locations and times. Figure 4 shows typical potential issues that could impede center point detection, even for experienced human operators. In certain instances, the leaves and branches of crops near the control points might obstruct the UAS line of sight, as depicted in (d) of Figure 4. Additionally, environmental factors like wind might lead to the accumulation of sand or dead branches on the squares, obscuring their features, similar to the example shown in (d) of Figure 4. Moreover, direct sunlight reflecting off the GCP material into the camera could result in sun glare, causing a blurred distinction between the white and black sections, as seen in (c) of Figure 4. Other confounding factors such as (limited) motion or focus blur can also occur.

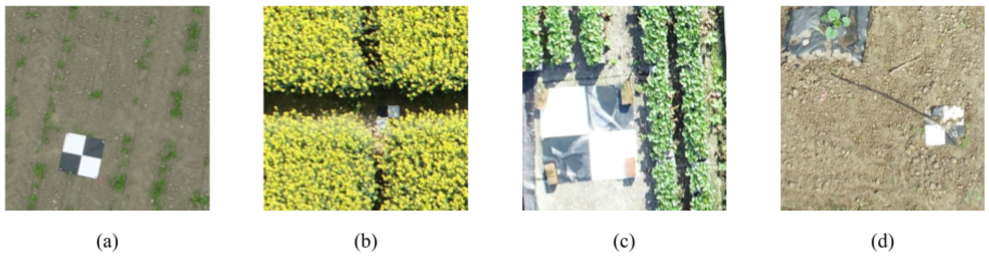


Figure 4. Examples of 512 × 512-pixel tiles, (**a**) a perfectly visible GCP, (**b**) GCP partially covered by the crop, (**c**) GCP reflecting sunlight, (**d**) GCP covered by sand.

2.2. Pre-Processing

Due to the dimensions and variable sizes of the entire individual images, training most deep learning models accurately for regression was infeasible. As a solution, the images were split into smaller, fixed sizes. A custom script was developed to extract tiles of a specified size encompassing the GCP center point. To prevent issues arising from tiles where the GCP center lay at the image edge, potentially impacting model accuracy, a padding parameter was implemented. This padding ensured that if a situation occurred where the GCP center was at the edge of a tile, an adjacent tile with better GCP visibility would cover it. Consequently, this technique could generate multiple tiles per GCP, depending on the padding amount.

The initial execution of the script, focusing solely on images with black and white control points, aimed to generate 512 × 512-pixel tiles with a 25% padding (128 pixels). This process resulted in 5658 tiles from the original 2980 images. Leveraging the human-labeled image pixel-coordinates for the GCP, the script efficiently determined tiles that undoubtedly contained a GCP. Moreover, the script provided the capability to include random tiles without associated labels, yielding a subset of tiles without any GCPs. The discussion section outlines potential applications for these empty tiles.

2.3. Data Augmentation

The diversity in drone flights offered a wide array of orientations of the ground control markers in the images, with various lighting characteristics as a result. The tiling algorithm employed on full-resolution images also introduced variability in the GCP position within each tile. The tiles covered instances of partially obscured control points, whether by sand, crop leaves, or material reflections. Initial testing revealed the model's subpar performance with tiles predominantly occupied by the GCP. This might stem from factors such as flight altitude, camera sensor specifications, resolution, and marker size. To address this issue and create more instances where the GCP covered the entire tile, an additional set of tiles, beyond the initially generated ones (512 × 512, 25% padding), was computed.

These new tiles featured a resolution of 224 × 224, aligning better with the requirements of the ResNet model without requiring resizing, and maintained the same 25% padding (56 pixels). This operation notably augmented the dataset by adding 5744 new tiles for training, validation, and testing. Figure 5 illustrates an image that generated two 512 × 512 tiles, as the GCP center lay within the padding area along one side. Conversely, the generation of 224 × 224 tiles resulted in four distinct tiles since the GCP center was situated within a corner within the 25% padding.

In essence, the marker predominantly occupied a larger portion of the smaller resolution tiles, providing the model with examples featuring larger GCPs and entirely new GCP center locations. Notably, despite the example in Figure 5, the number of tiles produced by the tiling script remained unaffected by the tile size.

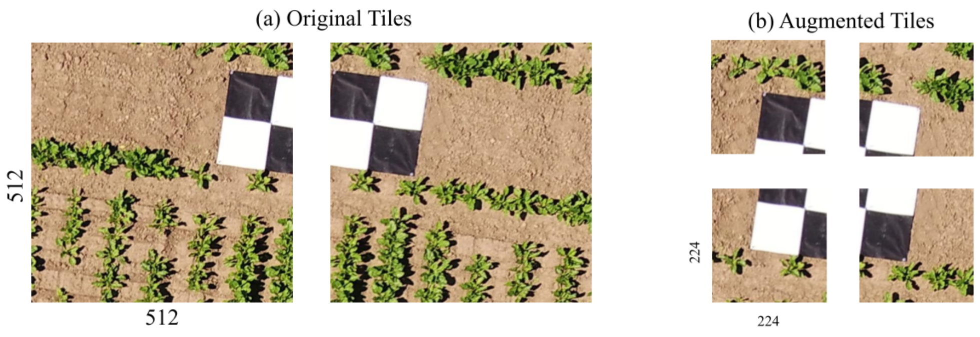


Figure 5. Comparison of the same ground control point tiled in 512 × 512-pixel tiles, as presented by figure (**a**) and tiled in 224 × 224 pixel tiles as presented by figure (**b**).

2.4. Computer Vision Approach

The determination of the edges of the ground control points involves a straightforward process based on the formation of four squares. This method was implemented using the CV2 Python library. Initially, the image was loaded (Figure 6a), and a binary mask was created to highlight the white pixels (Figure 6b). This process yielded two square clusters representing the white tiles of the GCP. The parameters for generating this binary mask are typically set between [255,255,255] and [240,240,240] (hexadecimal values ranging from *#FFFFFF* to *#F0F0F0*), which generally provided satisfactory outcomes.

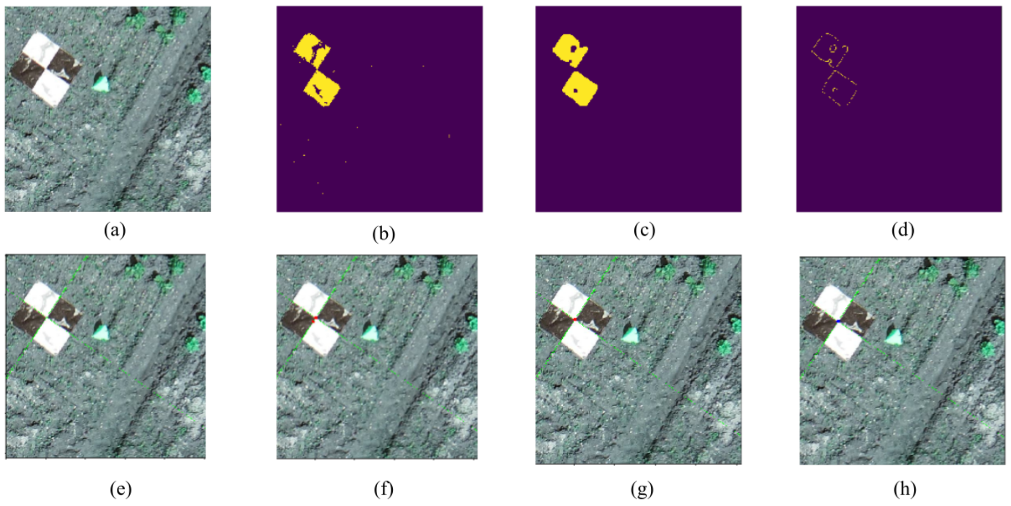


Figure 6. Visual representation of the computer vision pipeline, (**a**) original tile, (**b**) binary mask with *white* pixels, (**c**) binary mask after morphological operations, (**d**) Canny edge detection, (**e**) Hough Line Transform, (**f**) intersections of lines, (**g**) intersection of 90° angles, (**h**) average point.

To minimize noise interference in the sample, morphological transformations were applied to the mask to close internal gaps within the clusters. This step aimed to reduce the number of undetected pixels resulting from debris like sand or leaves covering the GCP. CV2 offers functions to perform dilation followed by erosion for this purpose.

The subsequent morphological transformation aimed to eliminate smaller clusters outside the ground control points, which could arise from factors like white stones or reflections in leaves. Utilizing CV2 functions for erosion followed by dilation, this operation retained only the essential clusters. Both transformations employed a specific kernel size, and adjustments to this kernel size would yield different outcomes. In the example, a 7 × 7 kernel was used (Figure 6c).

Following the isolation of the white tiles, detecting their edges ideally produces six segments forming two connected squares sharing a corner. This edge detection process employs the Canny edge detection algorithm [35] (Figure 6d). Modifiable parameters within these functions, notably two thresholds and the aperture size, allow for adjustments to attain optimal results.

The resulting image comprises two long segments intersecting in the center of the GCP, this task is well suited for line detection algorithms. The Hough Line Transform [36] (Figure 6e) operates by representing potential lines in the image as points in a two-parameter space: *Rho*, denoting the distance from the origin to the line along a normal vector, and *Theta*, signifying the angle of that normal vector. As the Hough transform algorithm computes all feasible lines that could intersect each edge in the image, it saves these lines in an accumulator matrix, a grid in the *Rho–Theta* space. The accumulation of votes in this matrix results in peaks that indicate regions where lines are most likely to exist in the image. It is imperative during this process to filter and isolate only the lines passing through the center of the GCP, disregarding those that intersect its edges.

After line detection, all intersections are computed and displayed as red dots in the image (Figure 6f). However, a challenge arises as the Hough Line Transform might detect multiple lines for the same GCP segment, leading to intersections distant from the GCP center. To resolve this issue, angles between each pair of lines are calculated. Only intersections originating from lines within a specified angle range, ideally close to 90º, are considered.

Eliminating intersections outside the [75º–105º] range refines the results (Figure 6g). Finally, averaging all the remaining points yields the outcome of this approach (Figure 6h). This calculated average represents the center of the GCP based on this methodology.

2.5. Deep Learning Models

In this paper, we tested three configurations of the ResNet model. By modifying the linear activation layer after the last bottleneck layer of the network with two output neurons, for both the coordinates of the keypoint, it is possible to train such models for regression. The two output neurons have a range of activation from 0 to 224 coinciding with the image-relative pixel coordinate of the center of the GCP after resizing both the tiles and the pixel coordinates to fit the input neurons of the model. The architectures tested for this study are ResNet-50, ResNet-101, and ResNet-152; these models are proven to be more accurate by considerable margins in classification tasks than smaller ResNets, such as ResNet-18 or even ResNet-34 [21]. All parameters in the ResNet architecture are trainable, and the number increases linearly with respect to the number of layers of the model. Table 2 shows these specifications as well as the size the fully trained models have.

This paper explores three configurations of the ResNet model by adjusting the linear activation layer post the final bottleneck layer. This modification involves incorporating two output neurons specifically tailored for the coordinates of the keypoint. These models are trained for regression, and the two output neurons are calibrated to have an activation range from 0 to 224. This range corresponds to the image-relative pixel coordinates of the GCP center, after both the tiles and pixel coordinates have been resized to align with the input neurons of the model.

Table 2. Size and number of parameters for residual neural networks utilized.

Model	Number of Parameters	Size of Model
ResNet-50	23,512,130	277.289 MB
ResNet-101	42,504,258	397.366 MB
ResNet-152	58,147,906	674.413 MB

The training pipeline was constructed using the PyTorch Python library. The architecture of the pretrained ResNets was sourced from the pretrainedmodels library available on GitHub (https://github.com/Cadene/pretrained-models.pytorch, accessed on 21 February 2024). This repository aims to provide access to pretrained Convolutional Neural Networks, aiding in the reproducibility of research paper results. Initial tests highlighted that utilizing pretrained models instead of training from scratch resulted in superior accuracy.

Prior research in image-related studies has consistently demonstrated the superiority of pretrained ResNets over their trained-from-scratch counterparts [37].

All models underwent pretraining using the ImageNet dataset—a repository comprising over 14 million images and accessible to researchers for non-commercial use [38]. The models were configured to accept torch tensors of 224 × 224 pixels and 3 RGB channels as input. Prior to training, the dataset was resized in memory to align with the required specifications of the models.

For the model evaluation process, three distinct loss functions were considered: mean absolute error (MAE) (Equation (1)), mean squared error (MSE) (Equation (2)), and smooth L1 loss (SL1) (Equations (3) and (4)). The smooth L1 loss function incorporates a default parameter beta, typically set to 1. This criterion employs a squared term if the absolute error falls below beta, otherwise utilizing the L1 norm. SL1 exhibits less sensitivity to outliers and, in certain scenarios, mitigates issues related to exploding gradients compared to MSE [39].

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{1}$$

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{2}$$

$$l(y, \hat{y}) = L = \{l_1, \ldots, l_N\}^T \tag{3}$$

$$l_i = \begin{cases} 0.5(y_i - \hat{y_n})^2/beta & \text{if } |y_i - \hat{y}_i| < beta \\ |y_i - \hat{y}_i| - 0.5 * beta & \text{if otherwise} \end{cases} \tag{4}$$

An experiment was conducted to assess the effectiveness of different loss functions used for training a single architecture. Three pretrained ResNet-50 models underwent training for 50 epochs, employing a learning rate of 0.001 and a batch size of 32. These models were trained using three distinct loss functions. Post-training, the models were evaluated against an independent test set, and their performance was assessed using all three loss functions to determine the most accurate model. The outcomes of this experiment are summarized in Table 3.

The model trained with mean absolute error (MAE) exhibited superior performance across all three evaluation metrics, achieving approximately a 35% enhancement compared to the model trained with smooth L1 loss (SL1). Notably, utilizing mean squared error (MSE) as the loss function for the ResNet resulted in the least accurate predictions among the three metrics assessed.

Table 3. Errors on predictions of the test set of models trained with different loss functions.

Model Trained with	Mean Absolute Error	Mean Squared Error	Smooth L1
MAE	0.858	1.405	0.481
MSE	3.074	21.338	2.606
SL1	1.237	2.788	0.831

The training pipeline employed the Adam optimizer [40], selected for its efficacy in optimizing stochastic objective functions based on adaptive estimates of lower-order moments. Notably, this optimizer requires minimal tuning and is ideal for first-order gradient-based optimization. Preliminary experiments indicated that all models converged before reaching the 50th epoch, signifying efficient convergence rates. Table 4 presents the shared hyperparameters utilized across all trained models in this study.

Table 4. Common hyperparameters utilized to train all models.

Hyperparameters	Value
Training epochs	50
Loss function	MAE
Training slip	≈70%
Validation slip	≈15%
Test slip	≈15%
Transfer learning	ImageNet
Activation function	Linear
Optimizer	Adam

All experiments were executed using Jupyter notebooks within the Google Colab platform. The allocated Graphics Processing Unit (GPU) provided by the platform was the *NVIDIA Tesla T4*, equipped with approximately 12.6 GB of Random-Access Memory (RAM) and 33 GB of available disk space. Additionally, the Central Processing Unit (CPU) allocated for computation was the *Intel Xeon CPU @ 2.20GHz*.

2.6. Experimental Setup

The complete raw dataset was utilized to generate two types of tiles: 224 × 224 and 512 × 512 pixels. These tiles were then distributed across a test set and a validation set, with each set containing 15% of the tiles. The remaining 70% constituted the training set. The primary focus of the experiments centered on identifying optimal hyperparameters for model training and comparing various ResNet architectures—ResNet50, ResNet101, and ResNet152. A total of 36 models were trained for the principal experiment, encompassing combinations of 3 learning rates ([0.01, 0.001, 0.0001]) and 4 batch sizes (16, 32, 64, and 96) across the 3 architectures.

During training, the validation split was employed to calculate the model's accuracy on each epoch based on the mean absolute error (MAE) criterion. Upon reaching a new minimum validation loss, the model was saved, ensuring that the stored model represented the instance with the lowest validation loss within 50 epochs.

Performance evaluation on the test sets was conducted using both mean absolute error and mean squared error metrics. The evaluation encompassed the complete test set, as well as test sets comprising only 224 × 224 and 512 × 512 tiles individually. Figure 7 visually illustrates the workflow of this study.

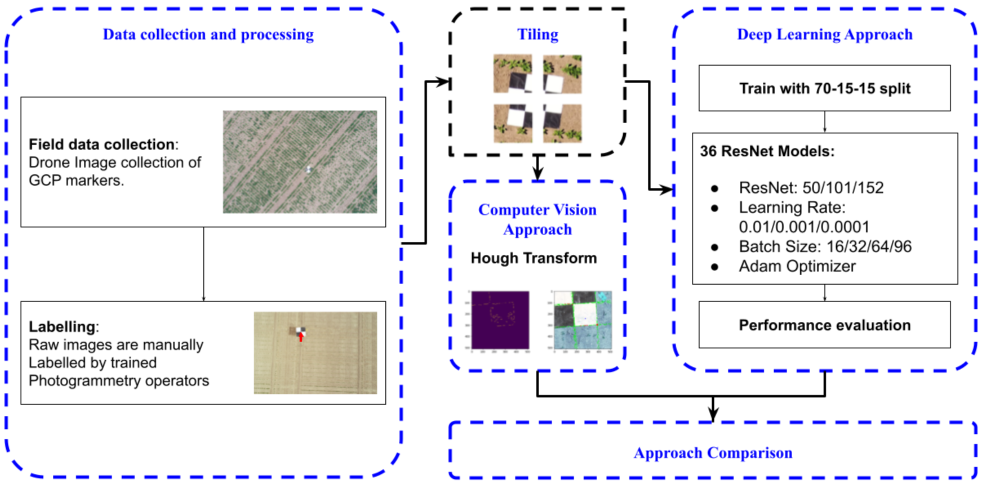

Data collection and processing
Field data collection: Drone Image collection of GCP markers.
Labelling: Raw images are manually Labelled by trained Photogrammetry operators
Tiling
Computer Vision Approach
Hough Transform
Deep Learning Approach
Train with 70-15-15 split
36 ResNet Models:
ResNet: 50/101/152
Learning Rate: 0.01/0.001/0.0001
Batch Size: 16/32/64/96
Adam Optimizer
Performance evaluation
Approach Comparison

Figure 7. Workflow of the main steps performed in this study.

3. Results

3.1. Computer Vision Approach

Initial testing revealed that the *Rho* parameter within the Hough Line Transform operator significantly impacted the results. In this function, the *Rho* parameter determines the granularity of the accumulator array in the Hough transform. A smaller *Rho* value results in a finer resolution of the accumulator matrix, potentially enhancing the detection of lines at finer resolutions and smaller gaps within the image. Conversely, increasing the rho value coarsens the resolution of the accumulator matrix, accelerating the algorithm but potentially limiting its ability to detect lines at finer resolutions or with subtle details. Figure 6 demonstrates a prediction with an absolute distance of approximately 1.96 pixels, specifically observed with a *Rho* value set to 1.15. Conversely, Figure 8 showcases three additional examples where the computer vision-based approach predicted the point within a reasonable distance from the human-labeled point. Notably, the optimal *Rho* parameter varied for each of these images and was specifically selected to optimize the predictions.

Figure 9 presents another tile computed using the same hyperparameters, as demonstrated in Figure 6. In this case, the Hough Line Transform operation erroneously detects lines along the outer edges of the white squares. This detection generates crossing points in all four corners, consequently shifting the average point away from the intended target point. This discrepancy results in a considerable difference of 75.56 pixels between the predicted and desired points. In more extreme scenarios, the Hough Transform may fail to detect any lines or produce angles between the lines that deviate significantly from 90°, leading to a lack of detected intersections and, subsequently, no point prediction.

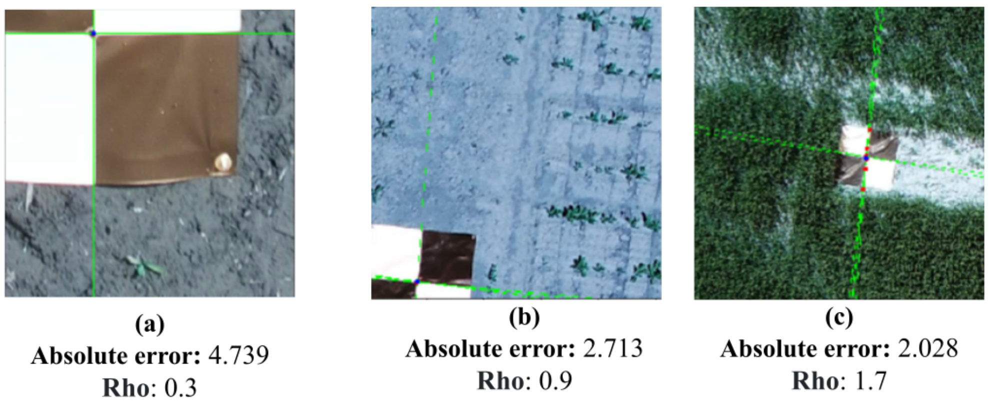

(a)
Absolute error: 4.739
Rho: 0.3
(b)
Absolute error: 2.713
Rho: 0.9
(c)
Absolute error: 2.028
Rho: 1.7

Figure 8. Visualization of the computer vision algorithm predicting the center points of different GCPs with the parameters utilized and the absolute error between the predicted point and the human-labeled point.

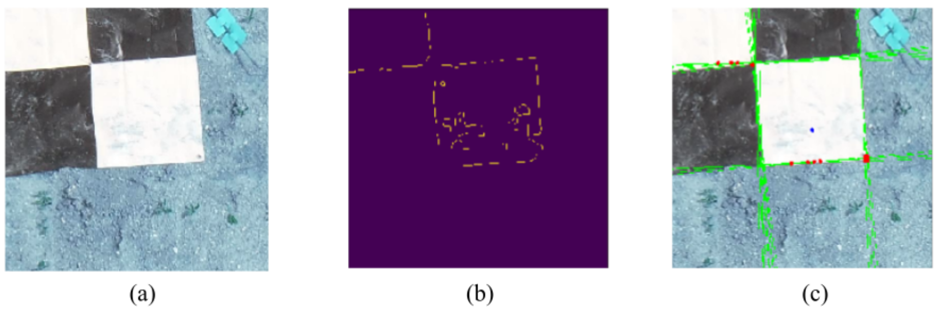


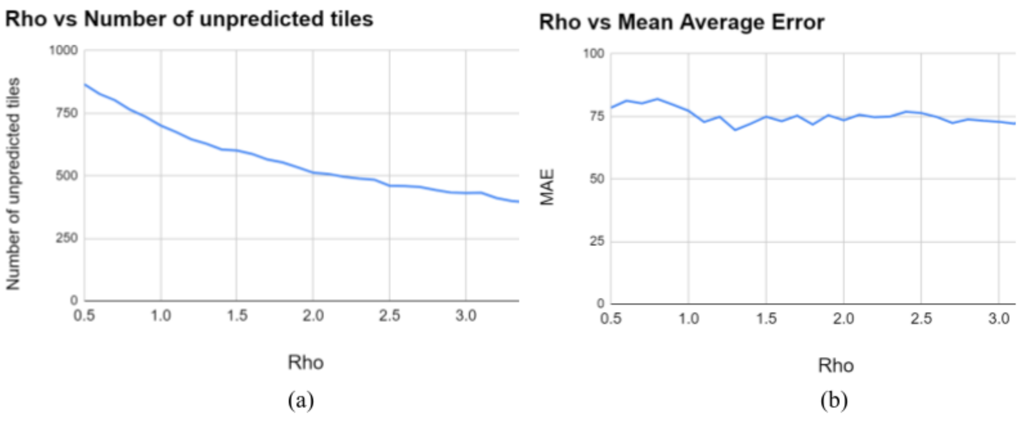
Figure 9. Example of the computer vision pipeline with non-optimal parameters for the tile (**a**) unaltered tile, (**b**) Canny edge detection, (**c**) crossing points, red points for intersections of lines, blue point as the average crossing point.

To investigate the impact of the *Rho* parameter on the Hough Transform and its effect on the computer vision approach's performance, an experiment was devised. This experiment involved generating predictions for 1000 512 × 512-pixel tiles while varying the *Rho* parameter from 0.5 to 3.5. Figure 10 illustrates the outcomes of this experiment across three graphs.

The first graph (a) in Figure 10 illustrates the number of unpredicted tiles against the *Rho* parameter. Lower *Rho* values result in fewer Hough Lines detected. Consequently, if no lines or their intersections have acute angles, no point prediction is made. With a starting *Rho* value of 0.5, over 85% of the images resulted in no prediction. However, as *Rho* increases, this proportion decreases, dropping below 40% at *Rho* = 3.5.

The second graph (b) in Figure 10 illustrates the Mean Absolute Error (MAE) for the computer vision pipeline against the *Rho* parameter. Variations in the parameter produce minimal changes in the MAE, maintaining a relatively consistent value of approximately 75 throughout the experiment.

Lastly, the third graph (c) in Figure 10 depicts the time required to predict the thousand-image batch against the *Rho* parameter. Beginning around 15 s with *Rho* = 0.5, the time exponentially increases with higher *Rho* values, reaching over 15 min with *Rho* = 3.5. Notably, there exists a substantial variance in computational time among individual images. Images that yield numerous lines necessitate extensive computational effort to compute all intersections between them.



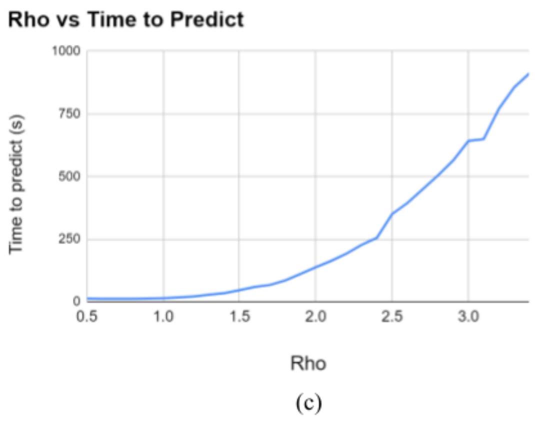


Figure 10. Graphs representing the results of testing the computer vision pipeline on a thousand images with different *Rho* values for the Hough Line Transform. (**a**) Number of unpredicted tiles vs. *Rho*, (**b**) MAE vs. *Rho*, and (**c**) time to predict 1000 images vs. *Rho*.

3.2. Deep Learning Approach: Influence of Tile Size

3.2.1. Influence with Training

The depicted graph in Figure 11 illustrates the outcomes of training using the chosen data augmentation technique compared to training solely with one tile size resolution. The model trained with a combination of tiles showcased significant performance gains. Specifically, when tested against 224 × 224-pixel tiles, it outperformed the rest by over 60%. Additionally, it showcased a 25% improvement with 512 × 512-pixel tiles and an exceptional 80% enhancement with the combined dataset of both resolutions. In contrast, the performance of the dataset with the smaller tile size was notably poor compared to the higher tile size or augmented datasets.

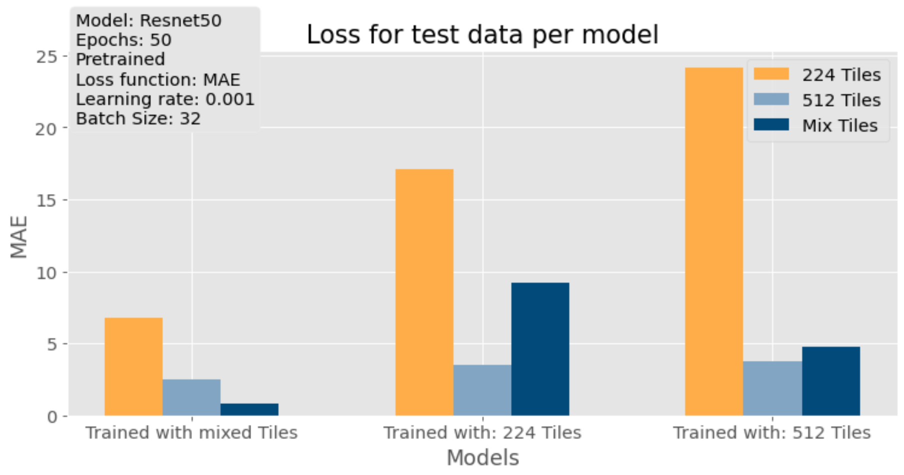


Figure 11. Mean absolute error of three ResNet-50 models trained for 50 epochs with: 224 × 224-pixel tiles, 512 × 512-pixel tiles and a mix of both. Learning rate of 0.001 and a Batch Size of 32. Results displayed against a test set containing only 224 × 224-pixel tiles (orange), 512 × 512-pixel tiles (light blue), and a mix of both (dark blue).

3.2.2. Influence with Testing

The graph presented in Figure 12 offers a comparative analysis of the Mean Absolute Error (MAE) results for all three architectures. These models were trained using a batch size of 32 and a learning rate of 0.001, utilizing test sets composed of 224 × 224-pixel tiles, 512 × 512 pixel tiles, and a combination of both, all sourced from the augmented dataset. Across the board, the results indicate that the models performed less accurately on the test sets comprising smaller tiles in comparison to the 512 × 512-pixel tiles and the augmented set. Notably, the test set involving mixed tile sizes consistently exhibited superior performance throughout all three models when compared to the other test sets.

In Figure 13, an extensive comparison of the Mean Absolute Error (MAE) across all trained models is presented against the three distinct test sets. These findings reinforce the trends observed in previous experiments. Specifically, the MAE scores from the 512 pixel test set were consistently, on average, 71.388% lower than those obtained from the 224 pixel test set.

Once again, the test set consisting of a mix of tile sizes demonstrated superior performance across all models. On average, its Mean Absolute Error was approximately 80.513% lower than that of the 224-pixel test set and exhibited a 30.113% reduction compared to the 512-pixel test set. These findings solidify the superiority of the model trained on mixed tile sizes in comparison to the single tile-size-based models.

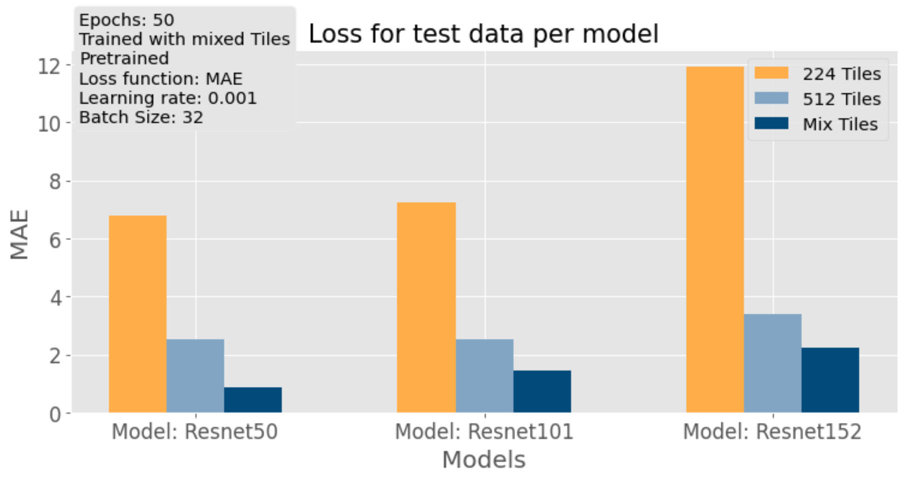


Figure 12. Mean absolute error of three models: ResNet-50, ResNet-101 and ResNet-152, trained for 50 epochs with a mix of 224 × 224- and 512 × 512-pixel tiles, with a learning rate of 0.001 and a Batch Size of 32. Results displayed against a test set containing only 224 × 224-pixel tiles (orange), 512 × 512-pixel tiles (light blue), and a mix of both (dark blue).

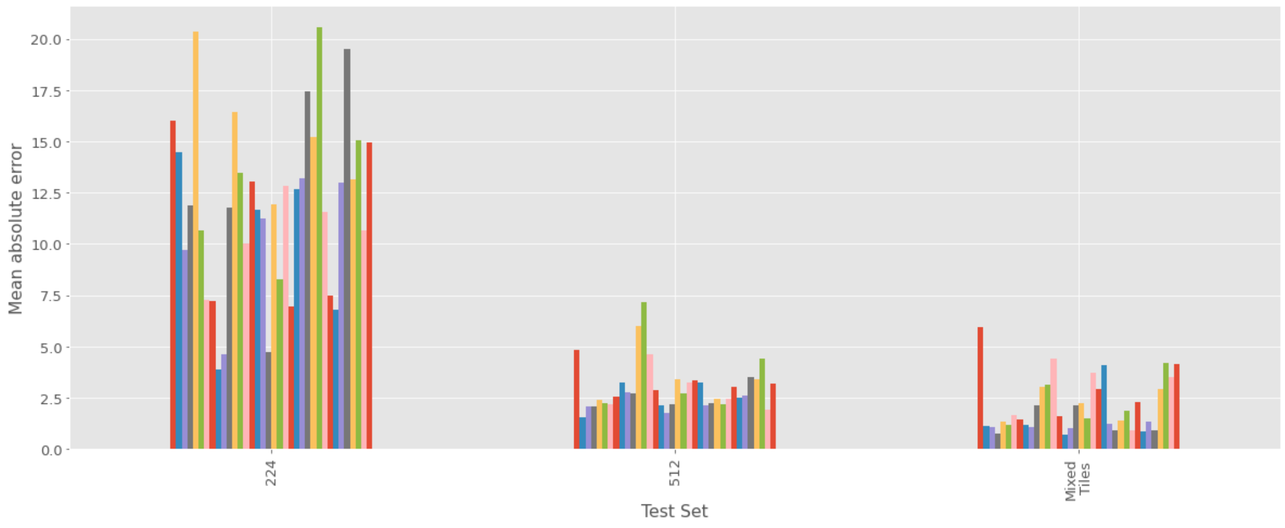


Figure 13. Comparison of mean absolute error of all trained models in order against a test set composed of 224 × 224-pixel tiles, 512 × 512-pixel tiles and a mix of both. Each color represents a single ResNet model with specific hyperparameters.

Table 5 provides an overview of the average MAE across different image-size test sets for each studied architecture (ResNet50, ResNet101, and ResNet152). Consistently, the data align with the previous observation highlighting the superior performance of the mixed tile models.

Across the architectures, performance is notably uniform, showcasing similar MAE values. However, ResNet152 exhibited superior performance in predicting center points within the 224 × 224-pixel tiles compared to the other architectures. In contrast, ResNet101 demonstrated relatively poorer performance in the 512 × 512-pixel tiles compared to its counterparts.

It is important to note that all trained models underwent testing against a diverse test set containing a mix of 224 × 224- and 512 × 512-pixel tiles.

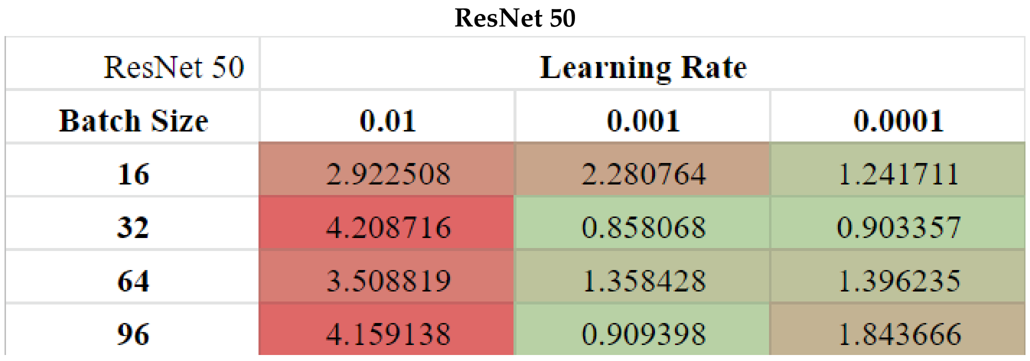
Table 5. Average mean absolute error across all trained models on different ResNet architectures with different test sets: 224 × 224-pixel tiles, 512 × 512-pixels tiles, and a mix of both.

Architecture	MAE for 224 × 224	MAE for 512 × 512	MAE for Mix Tiles
ResNet50	13.732	2.775	2.038
ResNet101	11.272	3.309	2.099
ResNet152	10.365	2.771	2.215

3.3. Deep Learning Approach: Main Results

Figure 14 outlines the performance outcomes for various batch sizes and learning rate combinations specific to ResNet50. Similarly, Figures 15 and 16 depict the corresponding results for ResNet101 and ResNet152, respectively. Notably, Figure 16 lacks results for models trained with a batch size of 96. This omission arises due to the substantial memory requirements (as detailed in Table 2) for ResNet152 models and the memory limitations of the Google Colab environment (approximately 12.6 GB), rendering the use of such a batch size unfeasible within this experimental setup.

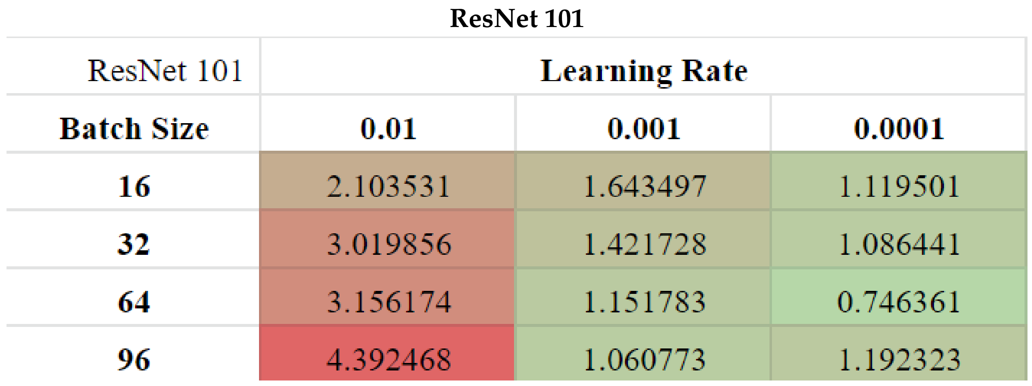
The impact of hyperparameter configurations on model accuracy is evident from the presented results. The most optimal performance was achieved by ResNet152, yielding an MAE of 0.723 when trained with a batch size of 32 and a learning rate of 0.0001. Following closely, ResNet101 attained an MAE of 0.746 with a batch size of 64 and a learning rate of 0.0001, securing the second-best performance. ResNet50, utilizing default parameters, obtained an MAE of 0.858 with a batch size of 32 and a learning rate of 0.001, positioning it in third place among the tested configurations.



ResNet 50

ResNet 50	Learning Rate		
Batch Size	**0.01**	**0.001**	**0.0001**
16	2.922508	2.280764	1.241711
32	4.208716	0.858068	0.903357
64	3.508819	1.358428	1.396235
96	4.159138	0.909398	1.843666

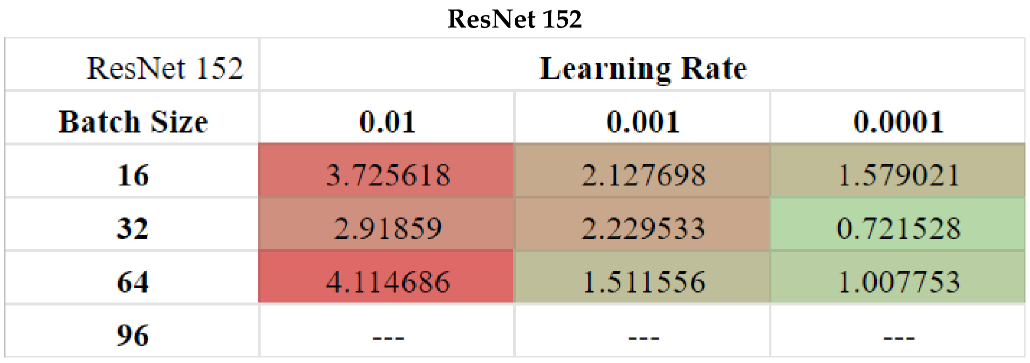
Figure 14. Accuracy (MAE) of ResNet50 with different hyperparameters (batch size and learning rate) after 50 Epochs of training.



ResNet 101

ResNet 101	Learning Rate		
Batch Size	**0.01**	**0.001**	**0.0001**
16	2.103531	1.643497	1.119501
32	3.019856	1.421728	1.086441
64	3.156174	1.151783	0.746361
96	4.392468	1.060773	1.192323

Figure 15. Accuracy (MAE) of ResNet101 with different hyperparameters (batch size and learning rate) after 50 Epochs of training.



ResNet 152

ResNet 152	Learning Rate		
Batch Size	**0.01**	**0.001**	**0.0001**
16	3.725618	2.127698	1.579021
32	2.91859	2.229533	0.721528
64	4.114686	1.511556	1.007753
96	---	---	---

Figure 16. Accuracy (MAE) of ResNet152 with different hyperparameters (batch size and learning rate) after 50 Epochs of training.

Table 6 shows the mean MAE and MSE across all three architectures as well as their standard deviations, while ResNet101 performed best overall, all models scored on average around 2 pixels MAE.

Table 6. Mean MAE and MSE as well as the standard deviation for the results of all trained models in all three architectures.

Architecture	Mean MAE	Standard Deviation for MAE	Mean MSE	Standard Deviation for MSE
ResNet50	2.133	1.265	12.061	12.915
ResNet101	1.841	1.116	9.180	10.439
ResNet152	2.215	1.172	14.032	14.473

The influence of the learning rate hyperparameter on the model performance is notably evident. As demonstrated in Table 7, reducing the learning rate correlates with enhanced accuracy. This table presents the mean values for both MAE and MSE, accompanied by their respective standard deviations, showcasing a consistent reduction in these metrics as the learning rate decreases. Similarly, Table 8 explores the impact of varying batch sizes on model performance, portraying MAE and MSE alongside their standard deviations. Interestingly, while mean values hover around 2 for MAE and 10 for MSE across different batch sizes, there appears to be no direct correlation between altering the batch size in isolation and achieving improved performance.

Table 7. Mean MAE and MSE as well as the standard deviation for the results of all trained models in all three studied learning rates.

Learning Rate	Mean MAE	Standard Deviation for MAE	Mean MSE	Standard Deviation for MSE
0.01	3.475	0.716	26.328	9.477
0.001	1.505	0.516	5.860	4.998
0.0001	1.167	0.341	2.463	1.437

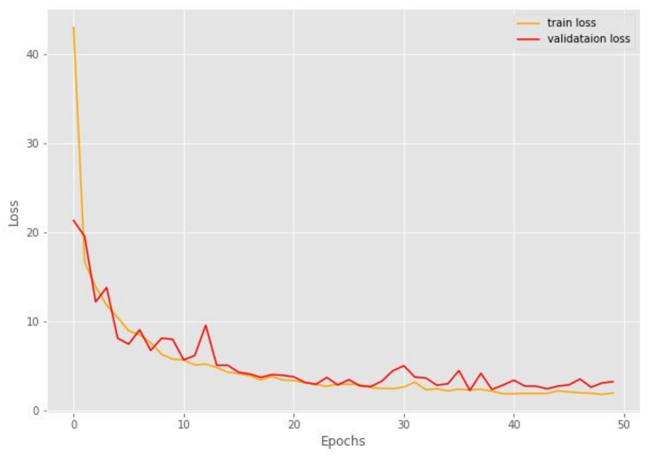
Table 8. Mean MAE and MSE as well as the standard deviation for the results of all trained models in all three studied batch sizes.

Batch Size	Mean MAE	Standard Deviation for MAE	Mean MSE	Standard Deviation for MSE
16	2.083	0.831	12.877	11.618
32	1.929	1.228	10.446	12.340
64	1.995	1.244	10.752	12.975
96	2.260	1.596	12.415	15.187

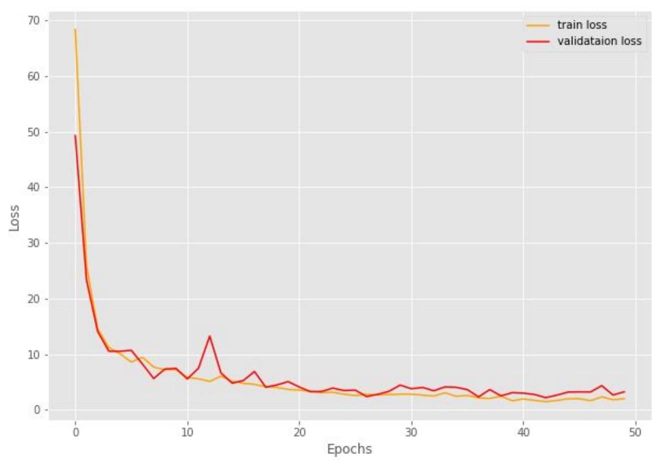
3.4. Deep Learning Approach: Individual Model Analysis

For validation of model convergence throughout training, both the training and validation MAE losses were recorded per epoch. Among these, the model chosen was the one displaying the lowest validation loss within the 50 epochs. Figure 17 visualizes the progression of training and validation losses across 50 epochs for each architecture. All models reach convergence well before the 35th epoch.

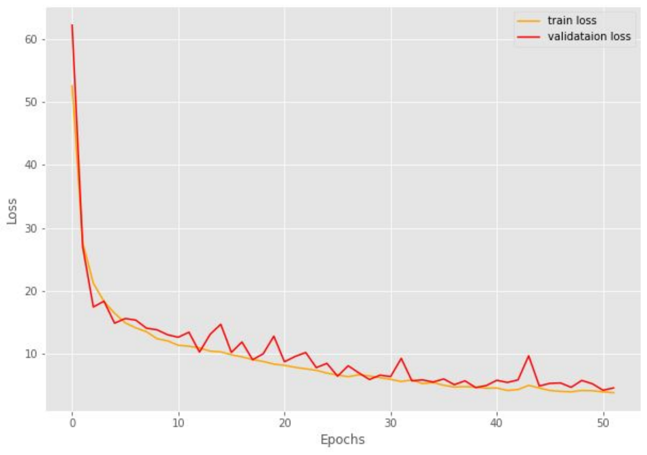
Displayed in Figure 18 are frequency distribution plots portraying the disparity between human-labeled and model-predicted X and Y coordinates. These plots focus on the three most effective models derived from the array of experiments conducted. A slight bias is noticeable towards a positive X coordinate (indicating predictions skewed to the right of the center point). Specifically, ResNet-50 model showcased an average X coordinate error of 1.14 pixels, while ResNet-101 displayed 0.33 pixels, and ResNet-152, 0.70 pixels. Correspondingly, their standard deviations were 6.19, 6.52, and 7.17, respectively. To address this bias, an experiment was executed, extending the training of these top-performing models by an additional 150 epochs each.



(a)



(b)



(c)

Figure 17. Graphs comparing the training loss MAE (orange) and validation loss MAE (red) on different models across 50 epochs of training. (**a**) ResNet 50: learning rate = 0.01, batch size = 64, (**b**) ResNet 101: learning rate = 0.001, batch size = 96 and (**c**) ResNet 152: learning rate = 0.01, batch size = 16.

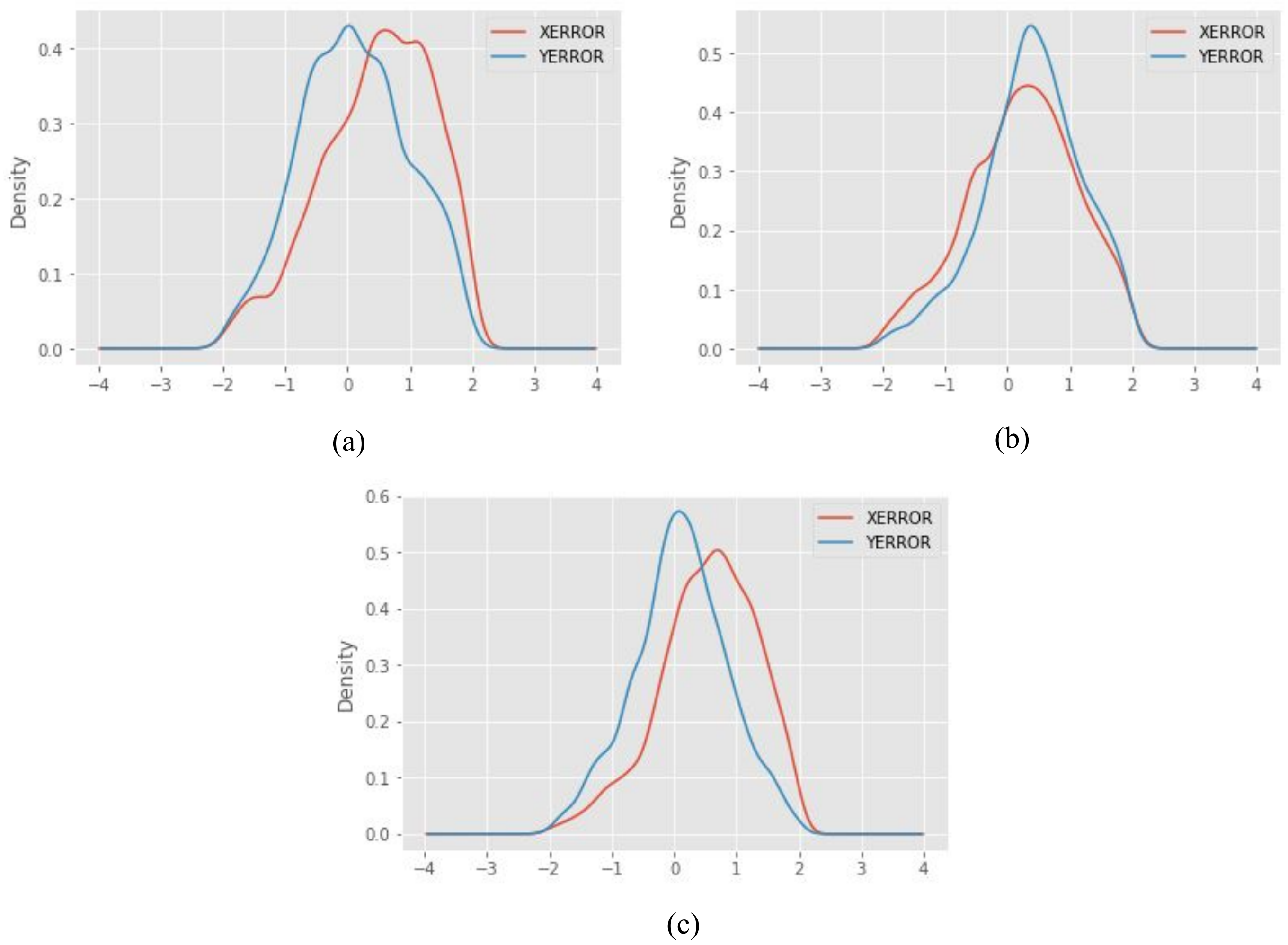
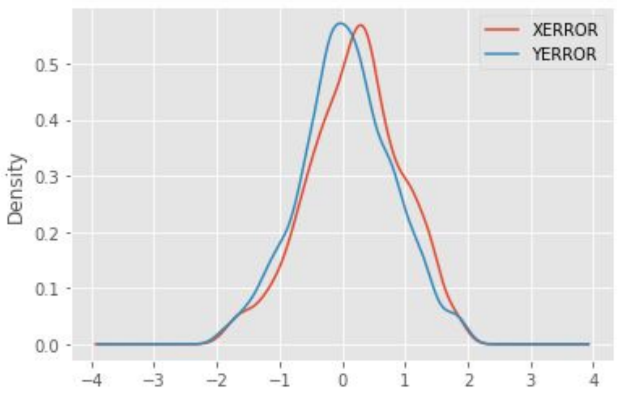


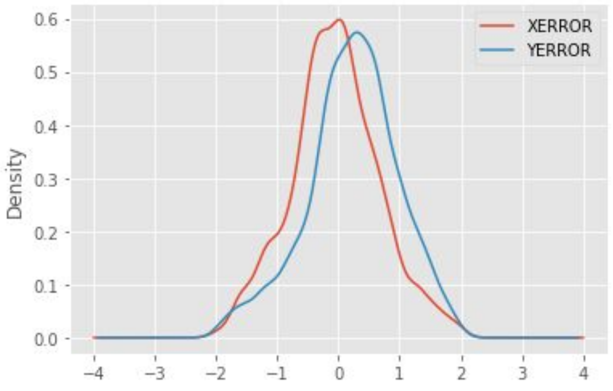
Figure 18. Frequency distribution plot for the difference between the human-labeled and model-predicted point coordinates (X red and Y blue). Three architectures trained for 50 epochs are displayed: (**a**) ResNet 50: learning rate = 0.001, batch size = 32, (**b**) ResNet 101: learning rate = 0.0001, batch size = 64 and (**c**) ResNet 152: learning rate = 0.001, batch size = 32.

In Figure 19, the outcomes of the extended training experiment are depicted. With the additional training, there is a marginal reduction in the mean absolute error (MAE) for all models, alongside a significant decrease in the bias toward the X coordinate. Post-extension, ResNet-50 exhibits an average error of 0.62 for the X coordinate, while ResNet-101 demonstrates −0.18, and ResNet-152 shows −0.06. Their respective standard deviations for this metric are 6.18, 9.18, and 9.06. In Table 9, the MAE for models after undergoing four times the initial epochs is tabulated, showcasing a notable reduction in losses across the board, with ResNet-152 achieving a value lower than 0.6.

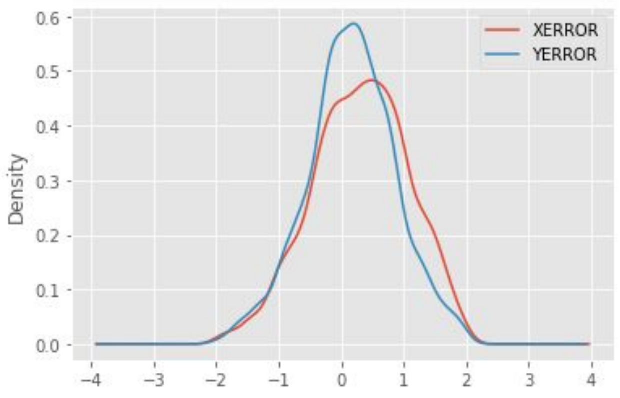
Figure 20 shows some examples of the top 50% predictions of the best performing model (ResNet-152: batch size = 32, learning rate = 0.0001 trained for 200 Epochs). The red dot represents the prediction of the model and the blue the human-labeled center of the GCP, if both points are in close vicinity the red one will appear on top. Predictions in problematic tiles are displayed in the examples, (b) and (d) Figure 20 show control points partially covered by leaves and sand. (a) Figure 20 shows a tile that is comprised almost entirely by the marker. (e) Figure 20 shows sun glare occurring, (c) Figure 20 is a marker whose center is on the edge of the tile and (f) Figure 20 is a smaller marker due to a higher resolution image.



(a)



(b)



(c)

Figure 19. Frequency distribution plot for the difference between the human-labeled and model-predicted point coordinates (X red and Y blue). Three architectures trained for 200 epochs are displayed: (**a**) ResNet-50: learning rate = 0.001, batch size = 32, (**b**) ResNet-101: learning rate = 0.0001, batch size = 64 and (**c**) ResNet-152: learning rate = 0.001, batch size = 32.

Table 9. Comparison of the MAE for the top three performing models after being trained for 50 and 200 epochs. (ResNet-50: learning rate = 0.001, batch size = 32, ResNet-101: learning rate = 0.0001, batch size = 64 and ResNet-152: learning rate = 0.0001, batch size = 32.)

Architecture	**MAE after 50 Epochs**	**MAE after 200 Epochs**
ResNet-50	0.858	0.664
ResNet-101	0.746	0.632
ResNet-152	0.721	0.586

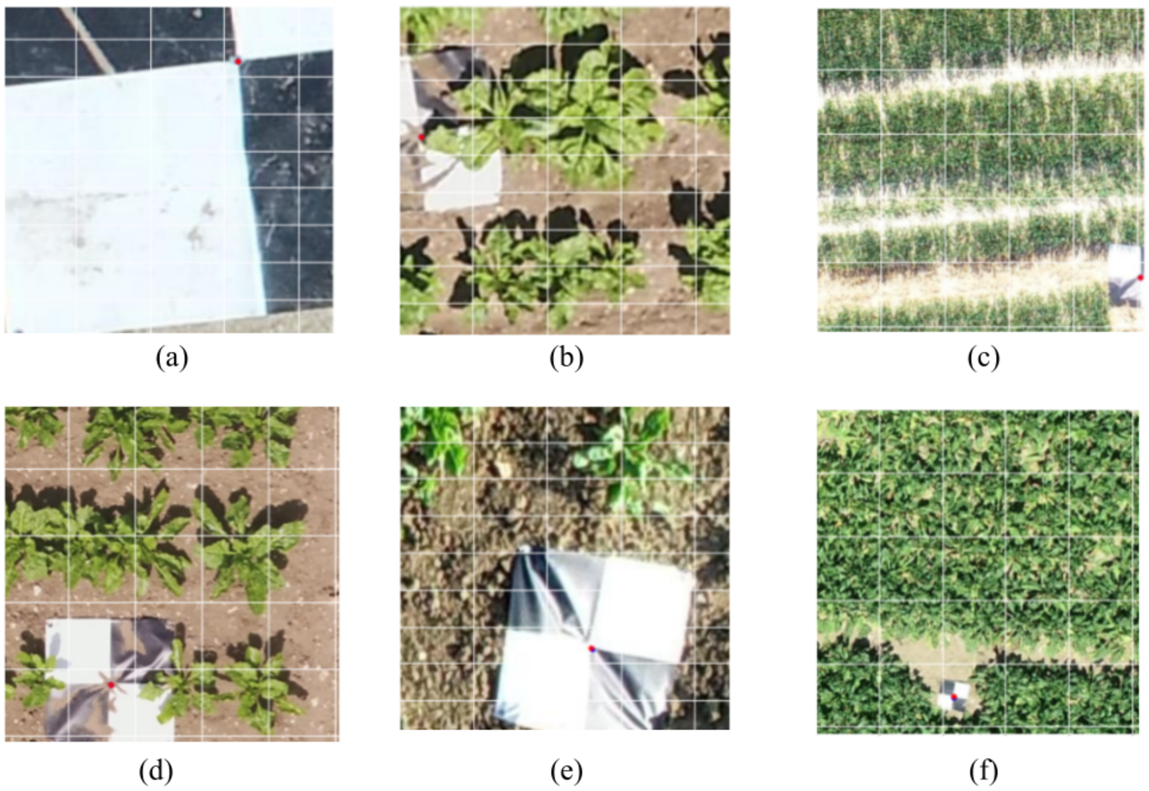


Figure 20. Examples of the top 50% predictions of the ResNet-152 trained for 200 epochs with batch size = 32 and learning rate = 0.0001. The blue point represents the human-labeled keypoint and the red point represents the prediction

In Figure 21, three examples of the worst 5% predictions are showcased to study the model's limitations. Instance (a) illustrates an extreme scenario with dense vegetation obscuring the tile, while in (b), environmental conditions and image acquisition parameters resulted in image blurring, making the center cross of the marker indistinct. In tile (c), the marker's center lies on the image's edge, leading to inaccurate center detection by the model. Conversely, within the bottom 5% of predictions, some instances exhibit the model's higher accuracy compared to the human-labeled center point. These cases are portrayed in Figure 22.

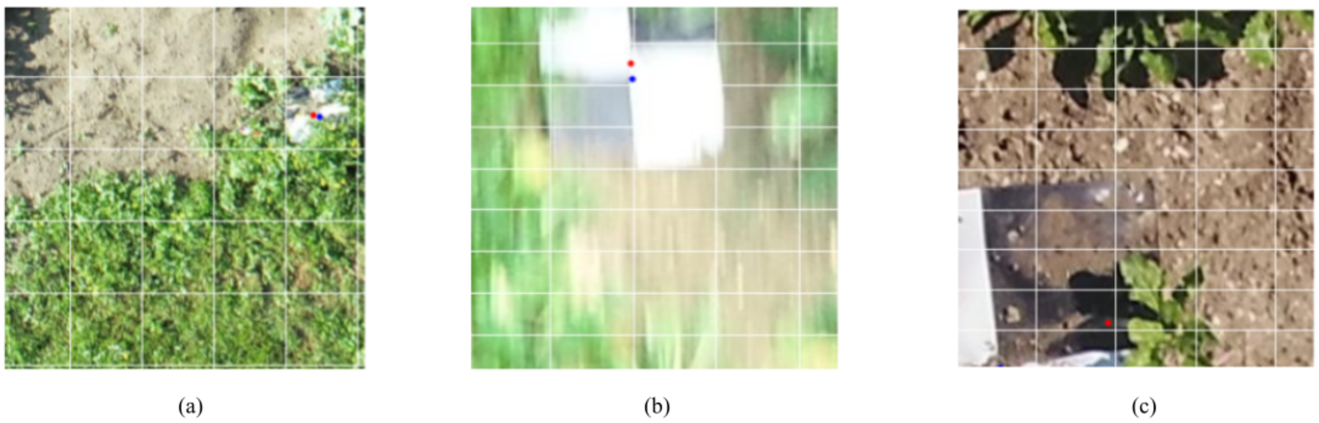


Figure 21. Examples of the bottom 5% predictions of the ResNet-152 trained for 200 epochs with batch size = 32 and learning rate = 0.0001. The blue point represents the human-labeled center point and the red point represents the prediction

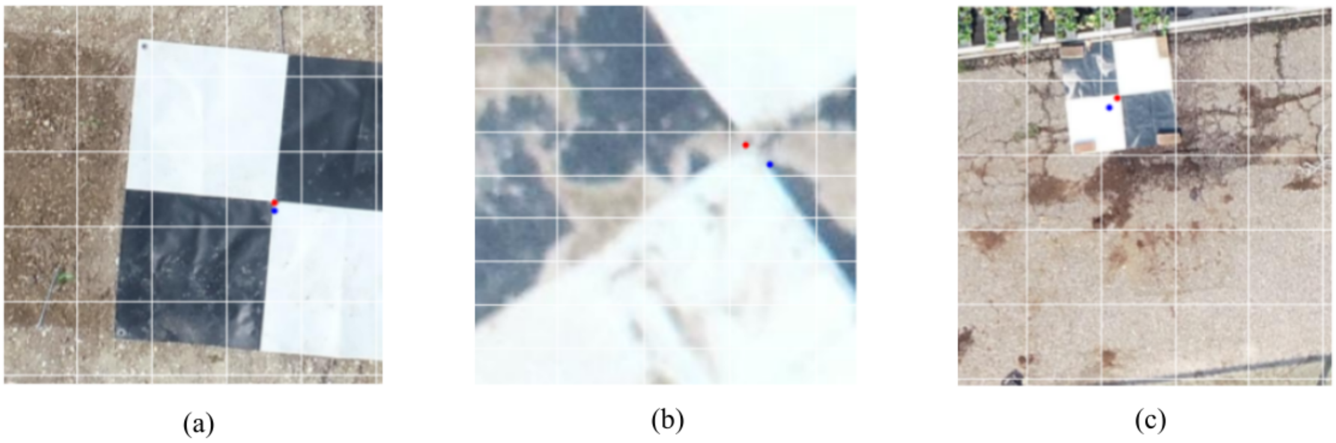


Figure 22. Three prediction examples with human error by the ResNet152 batch size = 32, learning rate = 0.0001, the blue point represents the human-labeled center point and the red point represents the prediction. Images (**a**,**c**) are 512 × 512-pixel tiles, and (**b**) is 224 × 224-pixel tiles.

3.5. Deep Learning: Training Time

Regarding the model training time, Table 10 shows the average time it took to train each of the different models for 50 epochs with all configurations. A deeper ResNet model means a higher number of parameters to train and backpropagate (Table 2) and, with this, a longer training time.

Table 10. Average training time for 50 epochs for all tester architectures.

Architecture	Average Training Time (s)	Average Training Time (HH:MM:SS)	Average Seconds per Epoch
ResNet-50	6559.116	01:49:19	131.182
ResNet-101	9802.033	02:43:22	196.040
ResNet-152	11,893.568	03:18:13	237.871

3.6. Deep Learning: Prediction Time

To test the inference time of the different models, a number of test set (1200 tiles) predictions were carried out. Table 11 shows the results of computing the center of the control points utilizing only the CPU (*Intel Xeon CPU @ 2.20GHz*) and utilizing also a GPU (*NVIDIA Tesla T4*). Similar to the training times, the deeper the model, the longer it takes for the inference. The inference time when using GPU is less than 3% of the inference time using only CPU.

Table 11. Average time of prediction of 1200 tiles with *batch size* = 32 for different architectures with only a CPU and with the use of a GPU.

Architecture	Average Seconds CPU	Average Min:Sec CPU	Average Seconds GPU	Average Min:Sec GPU
ResNet50	238.083	03:58	6.687	00:07
ResNet101	401.205	06:41	7.482	00:07
ResNet152	582.306	09:42	9.415	00:09

4. Discussion

This study aimed to explore a deep learning-driven methodology for accurately localizing the center of black and white squared Ground Control Points (GCPs) from RGB images captured by drones. We conducted a comparative analysis involving three ResNet architectures (ResNet-50, ResNet-101, and ResNet-152) with varying hyperparameters.

The ResNet-152 model, trained with a learning rate of 0.0001 and a batch size of 32, demonstrated the most favorable performance, achieving a mean average error (MAE) of 0.586 pixels, indicating sub-pixel accuracy. Comparable results were observed with other architectures, with the ResNet-50 and ResNet-101 achieving MAEs of 0.664 pixels and 0.632 pixels, respectively.

Studies relying solely on a computer vision approach, employing methods like the Hough transform, have demonstrated a precision level averaging 0.51 pixels in discerning centers compared to reference centers [36]. However, these methodologies were executed on small 12 $\times$ 12 pixel tiles exclusive to markers. Implementing such an approach in real-world scenarios necessitates an initial algorithm to accurately extract these centers, a method not detailed by the authors of this study. Without precise extraction of markers from images, this approach might lack applicability in diverse drone imagery contexts.

In our investigation, we experimented with a computer vision pipeline leveraging the Hough transform, akin to the reference study [36]. Unlike the referenced methodology, our pipeline processed complete 512 $\times$ 512-pixel tiles encompassing substantial background along with the Ground Control Points (GCPs), while this algorithm yielded reasonably accurate outcomes, its heavy reliance on parameter settings poses a significant challenge. Fine-tuning parameters becomes imperative for individual images, accommodating variations in lighting conditions, material coverage over control points and resolution disparities, among other factors. Consequently, this approach lacks the capability to generalize for automation. Modifying parameters such as the *Rho* value in the Hough Line Transform does not enhance overall performance but substantially reduces the number of unpredicted tiles at the expense of exponentially elongated prediction times for the pipeline.

An alternative study introduced an approach using edge-oriented histograms in conjunction with a modified Canny edge detection, coupled with a CNN [14]. This method effectively segmented GCP markers, accommodating variations in scale, rotation, and illumination. However, the pipeline demonstrated shortcomings when faced with image irregularities, such as unclear marker visibility, the presence of other white objects proximate to or beneath the GCP, and the influence of shadows on the GCP.

Studies using deep learning for broad GCP location and computer vision for center detection [16] are effective within the scope of controlled conditions. However, this method's reliance solely on computer vision algorithms for edge detection is susceptible to failure in scenarios less conducive to ideal conditions, as evidenced by the findings in this study.

In contrast, the deep learning approach adopted in our study exhibits robustness against these data irregularities (refer to Figure 20). By training the model with less distinct examples, it gains the capacity to discern GCPs across a wide spectrum of scenarios, offering enhanced adaptability to various environmental conditions.

The augmentation of the dataset through the inclusion of varied tile sizes has proven to be a potent strategy in enhancing the accuracy of diverse models. Upon comparing the performance of ResNet architectures (ResNet-50, ResNet-101, and ResNet-152), it is evident that all these architectures are capable of achieving sub-pixel Mean Absolute Error (MAE) against an independent test set, provided their hyperparameters are appropriately tuned.

While ResNet-152 emerged as the top-performing model after 50 epochs of training, achieving a modest increase of approximately 2.5% in accuracy over the best ResNet-101 model, this improvement comes at the expense of a larger model size (refer to Table 2). Moreover, employing ResNet-152 entails roughly 20% longer training times (as indicated in Table 10). Additionally, the computational resources required for predictions are notably impacted by deeper models; transitioning from ResNet-101 to ResNet-152 architecture increases the prediction time by approximately 30% when relying solely on CPU computations, although this effect is significantly mitigated when utilizing a GPU.

Choosing the appropriate model for a specific pipeline should consider the technical constraints and logistical implications of the project, while 50 epochs generally suffice for model convergence, further training can marginally enhance accuracy while reducing biases in error along any axis.

It is straightforward to reproduce the methodology delineated in this study concerning data preprocessing, augmentation, and machine learning predictions. Paramount to the successful deployment of this framework is the acquisition of an extensive, human-annotated dataset meticulously capturing the myriad of scenarios anticipated in real-world applications. Variables such as drone flight altitudes, diverse lighting conditions, GCP marker characteristics and potential obstructions on GCPs among other factors stand as pivotal components necessitating representation within this dataset. The comprehensiveness and fidelity of this dataset significantly underpins the robustness and adaptability of the proposed pipeline to diverse real-world conditions. Further, adequate finetuning of the hyperparameters can also significantly influence model performance.

This study exclusively focused on black and white-squared GCPs, leaving scope for further exploration regarding the transferability of the model to recognize other shapes and materials of GCPs. The models were specifically trained to detect the image-relative pixel coordinates of GCPs within tiles of limited, fixed size. Future investigations should delve into methodologies to identify the pixel coordinates of GCP centers within variably-sized entire images.

It is generally preferred by drone operators to fly in clear sky conditions, however due to weather and time constraints this isn't always possible. Stable conditions are otherwise preferred, to limit the appearance of clouds on the final orthomosaic. Limited instances of spotty cloud directly over GCP targets occurred in the dataset, and so performance in this regard has not been evaluated. For the purpose of model training, additional data augmentation to synthetically alter illumination can lead to better generalization in differing flight conditions.

Potential avenues for exploration could involve integrating a classifier to identify tiles containing GCPs and subsequently performing keypoint regression exclusively on those tiles. Alternatively, employing object detection on entire images to focus tiling efforts solely on areas where GCPs are detected could be explored. Further research could explore the utilization of the You Only Look Once (YOLO) architecture [41], a state-of-the-art, real-time object detection system. YOLOv7, in particular, has demonstrated high accuracy and speed across various scenarios [42], and its applicability for pose estimation via keypoint detection makes it a promising avenue to explore. Investigating a rapid, single-step approach to detect both the GCP marker and its center point using this model could be a valuable pursuit. Additionally, recently released multimodal large language models (LLMs) such as GPT-4V, Mixtral 8x7B and CogVLM are worth exploring for their capabilities in GCP center point detection, based on image tiles small enough to be ingested for inference. The same models can potentially be used to first look for GCP markers in entire images before tiling and selecting the relevant tiles for center point inference. Particularly, investigating improvements based on Set-of-Mark prompting using output from segmentation such as Segment Anything Model (SAM) holds potential to fast-track this process, especially as the hardware requirements of running inference based on LLMs can still present a bottleneck.

In practice, the model developed in this study will be integrated into an operational workflow. Its predictions will be cross-validated against further human annotations. The ultimate goal is to automate the generation of accurate, geo-referenced spatial datasets, rendering the process more precise and faster than manual intervention. This move towards automation seeks to replace human annotators, ensuring efficient and accurate processing of spatial data.

5. Additional Content

Tile Classifier

In this paper, a method to detect the centers of black and white squared GCP markers is outlined, albeit not encompassing the entire pipeline required for handling variable resolution images. As a proof of concept, a machine learning tile classifier was developed to differentiate tiles containing GCPs from those without.

Due to time constraints and this aspect not being the primary focus of the study, Microsoft Lobe was utilized to construct the model. Lobe is a desktop application facilitating the creation, management, and utilization of custom machine learning models for image classification. Training data, obtained through the tiling script, consisted of approximately 10,000 512 $\times$ 512-pixel tiles, with 60% devoid of GCPs and 40% including them. The application offers real-time control over the training process. Specifically, the model within Lobe can be optimized for speed, in which case MobileNetV2 is used, or for accuracy, in which a ResNet50-V2 is trained. Lobe automatically takes care of data augmentation in the background to improve training accuracy. Once an acceptable accuracy level is achieved, the model can be tested within the application by introducing new, unlabeled examples. Additionally, misclassified examples can be utilized for further training. Although the model was trained exclusively using 512 $\times$ 512-pixel tiles, Lobe supports training with variable-sized images.

Once the model was trained, it was exported as a protobuf file for integration with TensorFlow in Python3, enabling its incorporation into the broader pipeline. The Lobe desktop application purported a 98% accuracy in classification, albeit when tested against tiles it had already encountered. To validate its performance further, a new evaluation was conducted involving approximately 7000 unseen 512 $\times$ 512-pixel tiles—60% devoid of GCPs and 40% containing them. Table 12 demonstrates the outcomes of this assessment, indicating a close to 97% accuracy in classifying unseen tiles, both those with and without GCPs.

Table 12. Results of the Lobe trained classifier.

Statistic	Percentage
GCP classified as GCP	96.8034%
GCP classified as Empty	3.1965%
Empty classified as Empty	96.5290%
Empty classified as GCP	3.4709%
Total Accuracy	96.6847%

Based on the visual inspection of misclassified examples, it appears that many misclassifications occur when the GCP marker is positioned near the edges of the image, similar to the scenario depicted in image (c) in Figure 21. Additionally, shapes sharing structural and color resemblances with the markers may lead to erroneous classifications.

For a comprehensive view of the tile detection process and an illustration of a misclassified tile, Figure 23 provides a detailed depiction. The image is initially divided into 512 $\times$ 512 pixel tiles, each of which passes through the TensorFlow model. Tiles tinted in blue represent those identified as GCP-containing tiles. In one instance, the entire GCP marker is correctly identified within a single tile, while the surrounding tiles, despite padding, do not encompass the marker's center, correctly classified as non-GCP. However, an adjacent tile contains a signpost casting a shadow that bears a striking resemblance to a black corner of a GCP, potentially leading to the misclassification observed.

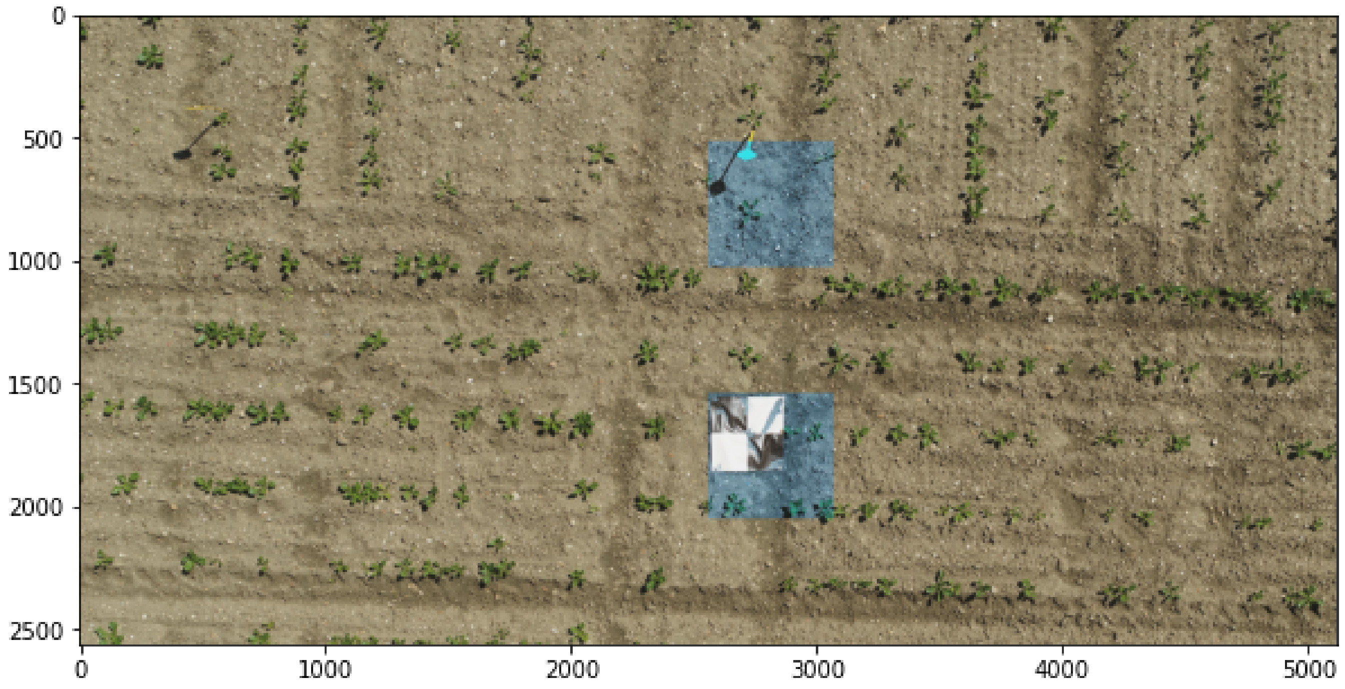


Figure 23. Example of the classifier on a full size image. The blue tinted tiles are the ones classified as containing a GCP.

6. Conclusions

This study extensively explored various ResNet architectures and their hyperparameters to accurately localize the center of black and white squared GCP markers within fixed-sized image tiles. Comparisons between deep learning and a purely computer vision approach employing Hough Transform revealed a significant enhancement in performance with the deep learning method. The ResNet-152 architecture, trained with a batch size of 32, a learning rate of 0.0001, and utilizing the Adam optimizer, showcased, to the best of our knowledge, acceptable accuracy with sub-pixel MAE results.

Notably, the investigation into automating GCP detection in drone imagery via a purely deep learning approach is a novel contribution in the literature. The study's outcomes underline the potential of the proposed pipeline in minimizing human efforts required for georeferencing aerial images. However, while promising, more research is essential to further generalize and optimize this approach for real-world applications across diverse scenarios. Exploring other model architectures and LLMs may contribute to these goals.

Author Contributions: Conceptualization, K.P. and D.R.; prototyping, K.P.; methodology, K.P. and S.O.; software, G.M.O.; analysis and validation, G.M.O., K.P., S.O., and D.R.; data curation, G.M.O. and S.O.; writing—original draft, G.M.O.; writing—review and editing, G.M.O., K.P., and S.O.; visualization, G.M.O.; supervision, K.P., S.O., and D.R.; funding acquisition, D.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this article are not readily available due to restrictions from 3rd party data providers. Requests to access the datasets should be directed to sam.oswald@vito.be.

Acknowledgments: VITO Remote Sensing is acknowledged for providing access to annotated GCP datasets available on the Mapeo platform. Thanks to Matthew B. Blaschko for their guidance and support throughout this project.

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

References

1. Tiwari, A.; Dixit, A. Unmanned aerial vehicle and geospatial technology pushing the limits of development. *Am. J. Eng. Res.* **2015**, *4*, 16–21.
2. Liba, N.; Berg-Jürgens, J. Accuracy of Orthomosaic Generated by Different Methods in Example of UAV Platform MUST Q. *IOP Conf. Ser. Mater. Sci. Eng.* **2015**, *96*, 012041. [CrossRef]
3. Hruska, R.; Mitchell, J.; Anderson, M.; Glenn, N.F. Radiometric and Geometric Analysis of Hyperspectral Imagery Acquired from an Unmanned Aerial Vehicle. *Remote Sens.* **2012**, *4*, 2736–2752. [CrossRef]
4. Lowe, D. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157. [CrossRef]
5. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
6. Zheng, Z.; Xiao, L.; Changkui, S.; Yongzhi, L. UAV tilted images matching research based on POS. *Remote. Sens. Land Resour.* **2016**, *28*, 87–92. [CrossRef]
7. Zhang, H.; Wang, L.; Tian, T.; Yin, J. A Review of Unmanned Aerial Vehicle Low-Altitude Remote Sensing (UAV-LARS) Use in Agricultural Monitoring in China. *Remote Sens.* **2021**, *13*, 1221. [CrossRef]
8. Bay, H.; Tuytelaars, T.; Gool, L.V. Surf: Speeded up robust features. In *Computer Vision—ECCV 2006: Proceedings of the 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.
9. Landau, H.; Chen, X.; Klose, S.; Leandro, R.; Vollath, U. Trimble's Rtk and Dgps Solutions in Comparison with Precise Point Positioning. In *Observing our Changing Earth*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 709–718.
10. Alhamlan, S.; Mills, J.; Walker, A.; Saks, T. The influence of ground control points in the triangulation of Leica ADS40 Data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* **2004**, *35*, 495–500.
11. Lee, H. Ground Control Points Acquisition Using Spot Image-The Operational Comparison. *Int. Arch. Photogramm. Remote. Sens.* **2000**, *33*, 528–533.
12. Zhou, G. *Determination of Ground Control Points to Subpixel Accuracies for Rectification of Spot Imagery*; Indiana State University: Terre Haute, IN, USA, 1990.
13. Ren, H.; Li, Z.N. Object detection using edge histogram of oriented gradient. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 4057–4061. [CrossRef]
14. Jain, A.; Mahajan, M.; Saraf, R. Standardization of the Shape of Ground Control Point (GCP) and the Methodology for Its Detection in Images for UAV-Based Mapping Applications. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Las Vegas, NV, USA, 2–3 May 2019*; Arai, K., Kapoor, S., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 459–476.
15. Yazdani, A.; Aalizadeh, H.; Karimi, F.; Solouki, S.; Soltanian-Zadeh, H. Sub-pixel X-marker detection by Hough transform. In Proceedings of the 2018 25th National and 3rd International Iranian Conference on Biomedical Engineering (ICBME), Qom, Iran, 29–30 November 2018; pp. 1–6. [CrossRef]
16. Zhu, Z.; Bao, T.; Hu, Y.; Gong, J. A novel method for fast positioning of non-standardized ground control points in drone images. *Remote Sens.* **2021**, *13*, 2849. [CrossRef]
17. Harris, C.; Stephens, M. A combined corner and edge detector. *Alvey Vis. Conf.* **1988**, *15*, 10–5244.
18. Beaudet, P.R. Rotationally Invariant Image Operators. 1978. Available online: https://www.semanticscholar.org/paper/Rotationally-invariant-image-operators-Beaudet/b80deba9cce6ad3bc8f5624c4a151a64ee226f14 (accessed on 23 November 2023).
19. Barroso-Laguna, A.; Riba, E.; Ponsa, D.; Mikolajczyk, K. Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.
20. Dai, Z.; Huang, X.; Chen, W.; He, L.; Zhang, H. A comparison of CNN-based and hand-crafted keypoint descriptors. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2399–2404.
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
22. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
23. Grosse, R. *Lecture 15: Exploding and Vanishing Gradients*; University of Toronto Computer Science: Toronto, ON, Canada, 2017.
24. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Training Very Deep Networks. *arXiv* **2015**, arXiv:1507.06228
25. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
26. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
27. Srinivasan, K.; Garg, L.; Datta, D.; Alaboudi, A.A.; Jhanjhi, N.; Agarwal, R.; Thomas, A.G. Performance comparison of deep cnn models for detecting driver's distraction. *CMC-Comput. Mater. Contin.* **2021**, *68*, 4109–4124. [CrossRef]
28. Ryu, B.Y.; Park, W.N.; Jung, D.; Kim, S.-W. Landmark Localization for Drone Aerial Mapping Using GPS and Sparse Point Cloud for Photogrammetry Pipeline Automation. In Proceedings of the International Conference on Electronics, Information, and Communication (ICEIC), Jeju, Republic of Korea, 6–9 February 2022.
29. Becker, D.; Klonowski, J. Object Recognition of a GCP Design in UAS Imagery Using Deep Learning and Image Processing—Proof of Concept Study. *Drones* **2023**, *7*, 94. [CrossRef]

30. Cheng, C.; Yang, J.; Wang, C.; Zheng, Z.; Li, X.; Dong, D.; Chang, M.; Zhuang, Z. Automatic detection of aerial survey ground control points based on Yolov5-OBB. *arXiv* **2023**, arXiv:2303.03041.
31. He, H.; Qiao, Y.; Li, X.; Chen, C.; Zhang, X. Automatic weight measurement of pigs based on 3D images and regression network. *Comput. Electron. Agric.* **2021**, *187*, 106299. [CrossRef]
32. Ryou, S.; Perona, P. Weakly Supervised Keypoint Discovery. *arXiv* **2021**, arXiv:2109.13423.
33. Wu, S.; Xu, J.; Zhu, S.; Guo, H. A Deep Residual convolutional neural network for facial keypoint detection with missing labels. *Signal Process.* **2018**, *144*, 384–391. [CrossRef]
34. Lin, Y.; Chi, W.; Sun, W.; Liu, S.; Fan, D. Human action recognition algorithm based on improved ResNet and skeletal keypoints in single image. *Math. Probl. Eng.* **2020**, *2020*, 6954174. [CrossRef]
35. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [CrossRef]
36. Aggarwal, N.; Karl, W. Line detection in images through regularized hough transform. *IEEE Trans. Image Process.* **2006**, *15*, 582–591. [CrossRef] [PubMed]
37. Aquil, M.A.I.; Ishak, W.H.W. Evaluation of scratch and pre-trained convolutional neural networks for the classification of Tomato plant diseases. *IAES Int. J. Artif. Intell.* **2021**, *10*, 467.
38. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
39. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015. [CrossRef]
40. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
41. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
42. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.