Automatic counting of planting microsites via local visual detection and global count estimation

Ahmed Zgaren, Wassim Bouachir, and Nizar Bouguila

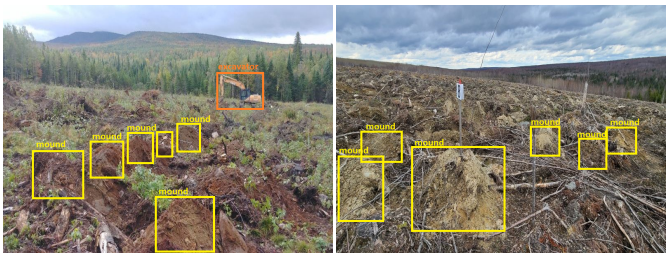
***Abstract*—In forest industry, mechanical site preparation by mounding is widely used prior to planting operations. One of the main problems when planning planting operations is the difficulty in estimating the number of mounds present on a planting block, as their number may greatly vary depending on site characteristics. This estimation is often carried out through field surveys by several forestry workers. However, this procedure is prone to error and slowness. Motivated by recent advances in UAV imagery and artificial intelligence, we propose a fully automated framework to estimate the number of mounds on a planting block. Using computer vision and machine learning, we formulate the counting task as a supervised learning problem using two prediction models. A local detection model is firstly used to detect visible mounds based on deep features, while a global prediction function is subsequently applied to provide a final estimation based on block-level features. To evaluate the proposed method, we constructed a challenging UAV dataset representing several plantation blocks with different characteristics. The performed experiments demonstrated the robustness of the proposed method, which outperforms manual methods in precision, while significantly reducing time and cost.**

***Index Terms*—Object counting, Object detection, computer vision, precision forestry, UAV imagery.**

I. Introduction

MECHANICAL site preparation by mounding is a popular technique often used by forest managers to ensure optimal growth conditions for tree seedlings. This procedure allows to address several soil problems that can severely impede root establishment of the planted trees, including high soil bulk density due to compaction by logging machinery, high water table, and plant competition for resources [1]. Mechanical preparation for planting is performed by using machinery (e.g. excavator) to create a mound corresponding to each planting microsite (see figure 1). Prior to planting operations, the number of planting microsites (mounds) present should be estimated, in order to determine the number of tree seedlings to be planted on each site.

The number of tree seedlings is often estimated by manual count, where several workers move through the field and visually identify each prepared mound on a portion of the site. The number of mounds is then estimated for the entire site by interpolating the partial result and assuming a constant density of mounds. This traditional counting method is time consuming and subject to errors. Moreover, the interpolation technique is not always reliable, as the density of mounds varies due to several factors, including site characteristics, preparation quality and type of machinery used. With the increased use of drones in the forest industry, some forest managers attempted to replace field work by photo-interpretation of UAV images.



(a) (b)

Fig. 1: Examples of mechanically prepared mounds in the balsam fir-white birch bioclimatic domain in Quebec, Canada. The fields present irregularities between different planting blocks, as well as between regions within the same planting block. The created mounds have different appearances, shapes, and sizes.

In this context, photo-interpretation consists of the detection of mounds by a human operator on a portion of the UAV image. Similarly to the fieldwork procedure described above, the final estimation is then predicted for neighboring regions, by interpolating the result. Although this method allows to significantly reduce the time required for field work, photo-interpretation is also subject to human error, in addition to imprecision due to terrain variability factors. These estimation errors often result in complex and imprecise handling of seedlings in the field, which causes monetary losses and additional planting delays.

Our work aims to propose an automatic process to fully take advantage of UAV imagery, coupled with computer vision techniques. We formulate the task of mound counting as a supervised learning problem based on the combination of two prediction models: 1) a local detection model used for detecting mounds based on deep features, and 2) a global estimation model for predicting the final count based on global features. The proposed system is trained in an offline manner to resolve two different, yet related, problems consecutively. On the one hand, the base object detector (local model) is trained to recognize visible mounds using a large dataset of manually annotated UAV images. On the other hand, the global estimation function is learned to predict the final count based on global information on the target field, including the detection result. Once the system is trained, online counting of mounds on a new image is performed by applying local and global models sequentially. More specifically, counting by detection is firstly performed using the local visual detector to

provide a preliminary counting of visible mounds. The preliminary count is then fused with global information extracted from the block's orthomosaic, to form a global feature vector. This information is finally used as the input of the global estimation function to obtain a precise prediction of mound count.

The proposed framework allows to automate the counting process by taking advantage of computer vision and UAV imagery. On the one hand, the use of our framework significantly reduces counting time, which currently takes several days to be accomplished using the manual method. On the other hand, we also improve counting precision, which has the potential to provide substantial financial benefits to forest managers.

We performed extensive experiments by testing the proposed method on UAV orthomosaics captured by overflying 12 planting blocks with different characteristics. These blocks are not included in the training set and contain 6164 mounds on average per block. The performed experiments demonstrate the high accuracy and robustness of the proposed method. With an overall relative counting precision of 95%, our computer vision solution outperforms both the manual field method as well as the photo-interpretation method. Further, we present an ablation study to investigate the contributions of each system module and several design strategies. Our analysis demonstrates the importance of different system components. In particular, it underlines that the combination of the two models (global and local) is more efficient than only applying a local visual detector. This is mainly due to the nontrivial nature of our object detection problem, with a high appearance variability at the scene level, and where objects of interest could be invisible due to several perturbation factors (e.g. occlusion by woody debris, water accumulation, mound erosion, and destruction).

Our main contributions can be summarized as follows:

- We propose a fully automated vision-based system to count mounds in orthomosaics. We thus contribute in addressing an important forestry management problem, by improving fieldwork conditions and significantly reducing time, money, and resource consumption for forest managers.
- We propose a robust computer vision framework, which can be generalized for a wide range of object counting applications, especially those using UAV imagery in crowded scenes.
- We construct a new dataset for mound detection and counting from UAV-based imagery.
- We analyse of the visual mound detection problem to identify the main challenges to be addressed in future work.

This paper is structured as follows. Section II, reviews relevant works from the literature on automatic object counting. In section III, we present the proposed method for mound detection and counting. Comprehensive experimental work is presented in section IV, including system evaluation in real-world conditions as well as an ablation study. In section V, we discuss the performance of our system with respect to several challenging situations. Finally, section VI concludes the paper.

II. Related works

The use of aerial imagery stimulated research work in several application areas related to forestry and agriculture, such as tree detection [2], animal counting [3], tree species classification [4], biomass estimation [5], and fire monitoring [6]. Several of these applications are based on the task of estimating the number of objects present in images, referred to as crowd counting. This section reviews the most important studies and methods proposed to count objects in crowded scenes. Methods from the literature can be categorized into three main approaches: 1) the traditional approach, mostly based on hand-crafted features and classical machine learning models, 2) the Deep Learning (DL) approach, where DL models are applied to learn and classify crowd regions, and 3) hybrid approaches, where both hand-crafted and deep features are used to improve the overall performance.

A. Crowd counting using traditional approaches

Traditional approaches are mostly based on image processing techniques to extract hand-crafted features [7]–[10], in addition to machine learning methods to predict object count. We can distinguish three main categories of methods: detection-based methods, where the final count is the total number of detected objects, regression-based methods, where a regression function maps the final count to the input features, and density estimation-based methods, where the final count is extracted from an estimated density map.

1) Detection-based methods: These methods generally adopt detection frameworks [11]–[14] by training a machine learning classifier on hand-crafted features. The used features include Harr wavelet [12], histogram of oriented gradient (HOG) [11], edgelet [13] and shapelet [14]. Detection is then performed either by global [11], [15]–[17], part-based [13], [18], [19], or shape learning [20] techniques. Several nonlinear classifiers have been also used to learn object patterns, such as support vector machine (SVM) [21], boosted trees [22] and random forests [23].

methods for counting by detection are generally successful in moderately crowded scenes but fail in handling highly-crowded scenes due to occlusions and scene clutter.

2) Regression methods: To overcome the dependency on learning detectors, regression-based methods aim to map the final count to input local features [24]–[26]. Models in this category are designed based on two main steps: low-level feature extraction and regression modeling.

Various types of features such as foreground features, edge features, texture, and gradient features have been used to extract visual information. While background subtraction techniques have been used to separate foreground features, Blob-based holistic features such as area and perimeter have also been successfully used to capture global properties of the scene [25]–[27]. To further increase accuracy, more advanced local features have been used, such as edges [28] and texture/gradient features [29], [30].

Once feature extraction is performed, different models can be used to learn a mapping function from low-level features to

crowd count, such as linear regression [31], piece-wise linear regression [27], and neural networks [32].

Idrinet al. [33] observed that there is no single feature or detector which is capable of providing sufficient information for precise counting in high-density scenes. They proposed to combine different feature extraction methods to capture multiple information types. Once local counts in all image patches are collected, they are fed to a global multiscale Markov Random Field (MRF) to estimate the final count.

Counting by regression mainly attempts to mitigate the former problems, like occlusion and scale variation, while improving crowd counting performance in highly crowded scenes. However, low-level feature representations are often unable to capture semantic information, and regressors tend to ignore the spatial information by regressing global count.

3) Density estimation-based methods: To handle the spatial information problem, Lempitsky et al. [34] introduced a model to linearly map between local patch features and their corresponding object density maps. They formulated density map learning as a minimization of a regularized risk quadratic cost function using a new loss function for learning. Motivated by the success of density map estimation, Pham et al. [23] proposed a nonlinear mapping model using random forest regression. They proposed a crowdedness prior to overcoming the problem of the large variation in appearance and shape, then trained two different forests corresponding to this prior. Wang et al. [35] proposed a faster method by clustering the feature space into subspaces and then by learning the embedding of each subspace formed by image patches. Xu et al. [36] observed that crowd density estimation methods are based on computationally expensive Gaussian process regression or Ridge regression models, which can only handle a small number of features. They proposed to incorporate a rich set of features to boost the performance of crowd density estimation.

Crowd counting by traditional methods was successful in moderately crowded scenes and the proposed models did not require significant computational resources. However, traditional methods are based on low-level features for image representation, which limits their capability to handle difficult situations like appearance variation, scale variation, and context information.

B. Deep learning for Crowd counting

During the last decade, deep learning models have outperformed traditional machine learning approaches in various application fields. Due to their deep and sophisticated architectures, DL models proved their ability to extract robust semantic information from large datasets, which allows handling multiple challenging situations such as scale change, rotation, and appearance variation.

Motivated by the success of deep neural networks in computer vision, researchers have attempted to solve the object counting problem in crowded scenes using DL models. These models are mostly based on Convolutional Neural Networks (CNNs) to learn a non-linear function from crowd images to their corresponding density maps or counts.

1) Basic CNN approaches: Since these methods are among the first deep learning formulations for crowd counting, they integrate basic CNN layers to the proposed models. Wang et al. [37] proposed an end-to-end CNN regressor for people counting. The model is based on AlexNet [38] architecture, where the final fully connected layer is replaced by a single neuron to predict the final count. Fu et al. [39] optimized a multistage CNN by removing some network connections according to the observation of the existence of similar feature maps. Then, two CNN cascade classifiers were designed to classify crowd images into five classes: very low, low, medium, high, and very high. To optimize the model on cross-scene crowd counting, Zhang et al. [40] proposed to alternatively train a CNN for two tasks: density estimation and crowd counting. To further improve results on a target scene, the model is fine-tuned using training samples similar to the target. To increase counting accuracy and time processing, Walach et al. [41] proposed an improved CNN architecture with layered boosting and selective sampling techniques during training. In [42], the entire image was fed to the network for count prediction, instead of dividing it into patches. The authors combine a CNN with a recurrent neural network (RNN) to take advantage of contextual information when predicting both local and global counts.

2) Scale-aware CNN approaches: More advanced CNN architectures were demonstrated to be robust to scale variation, by incorporating different techniques such as multi-column or multi-resolution networks [43]. To ensure robustness and scale invariance, Boominathan et al. [44] proposed to combine deep and shallow fully convolutional networks to capture semantic informer density map estimation and to ensure robustness to non-uniform scale and perspective variations. Zhang et al. [45] presented the Multicolumn CNN (MCNN), which consists of three columns for different scale capture (large, medium, small). Onoro and Sastre [46] developed a scale-invariant CNN model (HydraCNN). Firstly, they introduced the Count CNN (CCNN) architecture that incorporates the perspective information for geometric correction of the input features. Then, they constructed a network of three heads and a body. Each head is a CCNN architecture for learning features of a particular scale. Finally, the outputs of all the heads are concatenated and fed to the body to estimate the final density map.

Sam et al. [47] proposed a scale-aware crowd counting model based on switching CNNs. Their architecture consists of multiple MCNN regressors and a switch classifier trained to select the optimal regressor. Later, Ranjan et al. [48] proposed a multi-stage crowd count process (ic-CNN) to generate high-resolution density maps. The model architecture is based on stacking multiple ic-CNN, which consists of a two-branch CNN, where the first branch produces a low-resolution density map and the second merge the outputs of the first branch to produce high-resolution density maps. In a different approach to incorporate scale information, Zhang et al. [49] extracted feature maps from different layers of a backbone and combined them to produce the final density map. In [50] Jiang et al. proposed a Trellis Encoder-Decoder (TEDnet) for crowd counting. The model focuses on generating high-quality

density estimation maps. They modified the Trellis architecture to incorporate rich scale information by using dense skip connections across paths. Recently, Dong et al. [51] proposed an end-to-end scale-aware model (MMNET) that integrates multi-scale features generated by different stages, to handle scale variations. Hu et al. [52] estimate the density map using Neural Architecture Search (NAS), which exploits multi-scale features and addresses the scale variation issue in counting by density. Moreover, Liu et al. [53] proposed a multiscale parallel encoder by introducing an Efficient and Lightweight Convolution Module (ELCM) to extract different scale features. Then, a Scale Regression Module (SRM) decoder is used to generate the density map. More recently, Aldhaheri et al. [54] proposed to filter out background noise from foreground features using a segmentation guided attention mechanism.

3) Context-aware CNN approaches: Incorporating local and global contextual information into the CNN architecture for crowd counting is a complex task that has attracted several researchers. Sheng et al. [55] proposed to integrate semantic information by learning locality-aware feature (LAF) sets. The proposed architecture comprises three main components. First, a CNN transforms the input raw pixel high-resolutions attribute feature map. Then, following the idea of spatial pyramids on neighboring patches, the LAF is introduced to explore more spatial context and local information. Finally, the local descriptors from adjacent cells are encoded into image representations using the VLAD [56] encoding method. Moreover, Ilyas et al. [57] have designed an end-to-end CNN model for crowd counting. The architecture of the proposed method consists of two parts: a deep feature extraction network (DFEN), and a scale-aggregation module with dilated convolution (SAD). They combined DFEN and SAD to collect large-scale contextual information, handling the perspective distortion and expanding the spatial sampling location. Also, Tian et al. [58] combined Density Aware Network (DAN) and Feature Enhancement layer (FEL) to capture local and global contextual features. In a different training approach, Lei et al. [59] proposed a weakly-supervised density estimation model. The model was trained using primary and auxiliary brands and two different annotation types, location level annotation, and count level annotated images. Recently, Do [60] proposed to learn global context information using a visual transformer. Liu et al. [61] proposed a multi-task Encoder-Decoder density map generator to learn the counting of maize stand from UAV images. To alleviate the problem of density dependence, the authors proposed to incorporate count and density map errors into the loss function.

Crowd counting using CNN-based architecture has been successful in both moderately and highly crowded scenes. Following this direction, multiple network architectures have been introduced to handle challenging situations such as scale variations and occlusion.

C. Hybrid methods

Hybrid methods mainly combine two feature representations: hand-crafted features, and deep features. Features can be extracted explicitly or implicitly, to be used to train either a classical machine learning classifier or a deep neural network to count objects. Lin et al. [62] proposed a method for counting persons in videos when crossing a line for low-cost devices. The proposed architecture uses a knowledge distillation approach to transfer knowledge from the CNN object detector, in order to train a small Local Binary Patterns (LBP) cascade classifier. After YOLO [63] detects persons in video frames, images of persons with confidence higher than 30% are used to train the LBP cascade classifier, which is used to detect-and-track pedestrians. Bouachir et al. [64] proposed a two-stage detector to automate the estimation of the number of planting microsites from multispectral images. The first stage is a cascade detector based on LBP features to generate region proposals that are likely to correspond to objects of interest. The second stage consists of a CNN applied for candidate region classification. This method allowed to handle difficult situations where the objects are difficult to detect (e. g. appearance variability, motion blur). However, this method suffers from a high computational cost due to the use of a sliding window process and region proposals. Besides, hybrid methods in general may also suffer from high computational complexity due to the fusion of multiple features and the use of more than a single classifier.

In our work, we propose an hybrid framework where we adopt two different approaches sequentially. However, unlike the above discussed hybrid methods that often suffer from high computational complexity, we perform visual feature extraction only during the first detection stage. This allows to maintain a reasonable computational complexity that corresponds to that of the object detector. Further, our second prediction stage is inspired by regression methods in that counting is predicted globally, regardless of individual object localization. Nevertheless, we do not process any visual low-level features, as our regression function is trained on global properties describing an orthomosaic. As demonstrated further in our experiments, the proposed conception allows to significantly improve counting precision compared to merely using a state-of-the-art object detector.

III. Proposed method

A. Motivations and overview

The aim of our work is to design a method to accurately estimate the number of planting microsites (mounds) present on mechanically prepared planting blocks. The input of the proposed system is an orthomosaic representing a new planting block (as shown in figure 3), each orthomosaic being constructed from high-resolution UAV images. We can also see in figure 3 that visual recognition of mounds could be a complex task even for a human. This is mainly due to several challenges, such as the similarity in appearance between mounds (objects of interest) and surrounding terrain (background), shape/appearance variation between mounds (intra-block variability), and appearance variability between different planting blocks (inter-block variability).

To handle these difficulties, system training is done by constructing two prediction models consecutively. We first construct a visual object detector from manual annotations, to

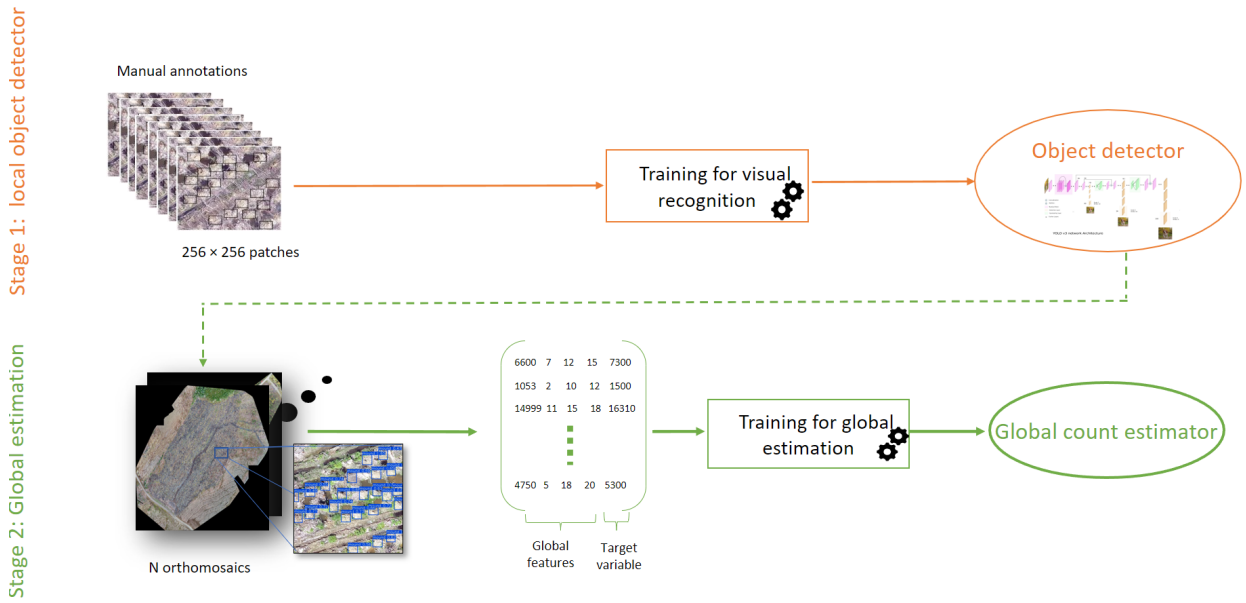


Fig. 2: System training. *Stage 1* illustrates the visual detector construction and *Stage 2* the global correction module training. Continued arrows indicate the transition between steps and dashed arrows indicate the use of a system component in the target step.

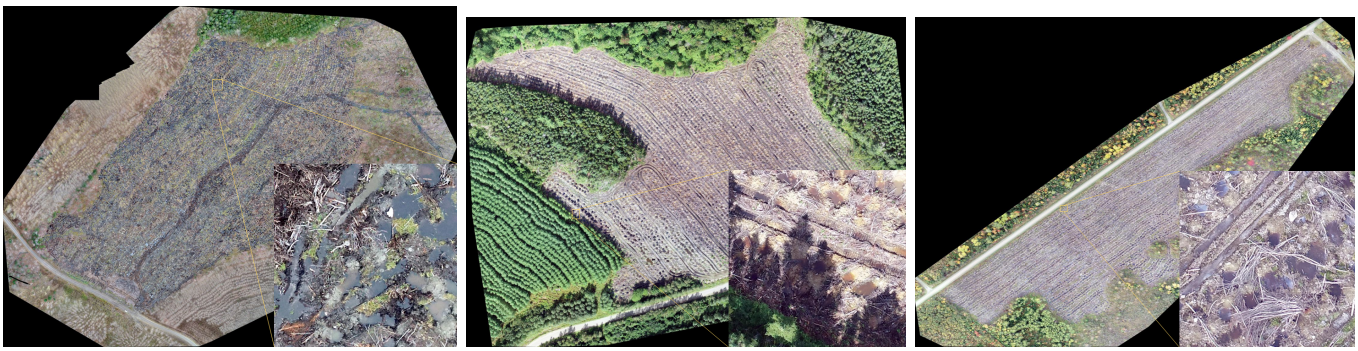


Fig. 3: Examples of orthomosaics captured and reconstructed for 3 different planting blocks. Corresponding areas are 13 hectares (ha), 4 ha, and 6 ha, respectively for (a), (b), and (c).

locally recognize visible mounds individually. Deep features are used in this first model for the visual representation of objects. In fact, hand-crafted features have been demonstrated to be limited for such a complex detection problem, which may result in a large number of false positives [64]. Further, we construct a global estimation model for predicting the number of mounds from the global features of a given planting block. The used feature set comprises block-level information, such as block area and prior knowledge on mound density. The system training process is illustrated in figure 2.

Once the entire system is trained, mound counting on an orthomosaic representing a new planting block is performed by applying the two models subsequently. That is, the local detection model is first applied to perform counting-by-detection, which is considered as a preliminary estimation of the number of mounds based on visual recognition. Note that local detection is preceded by a fine-tuning process to handle the inter-block variability problem mentioned above. This is achieved by providing the detection model with a few examples of mounds present on the new block. During the second stage, the counting-by-detection result is used, in addition to block-level information, as the input of the global estimation model to produce a final estimation of the number of mounds.

As explained above (and demonstrated further in the experiments), the only use of local object detection is insufficient, as local visual detection may be hampered by various perturbation factors, such as the presence of invisible mounds (e.g. occluded, destroyed) and confusion with the background texture. Both models, each using a different type of feature, are thus essential and complementary for accurate counting under our application constraints. Figure 4 summarizes the process of analyzing a new orthomosaic for mound counting. The proposed framework is detailed in the following sections.

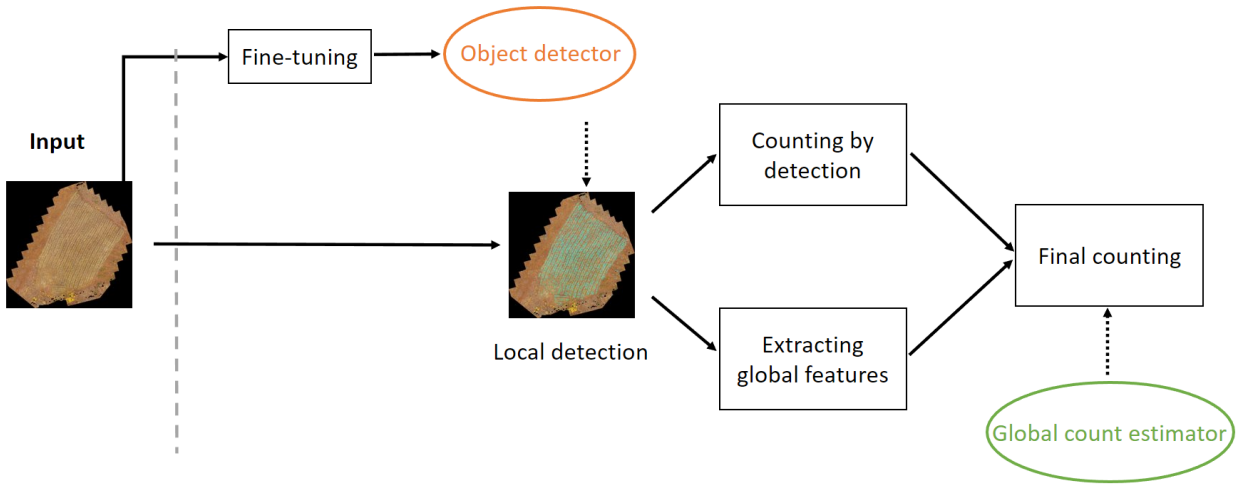


Fig. 4: Illustration of our framework while processing new orthomosaic. We fine-tune the trained object detector, then we infer patches to the detector for local mound detection. Once the number of detectable mounds is estimated, a feature vector is constructed by extracting global features. Finally, we apply the global estimator to predict a final count.

B. Local visual detection

In this work, we use YOLO [63] as a supervised one-stage approach for object detection. YOLO achieved state-of-the-art performance in several detection tasks while demonstrating robustness against scale and perspective variations [65], [66]. This method is built on the principle of merging very deep CNNs for feature extraction with multi-scale detection in a one-stage detection approach. As a result, the inference time for an image with a size of 608×608 pixels is about $50ms$ on a GPU device, with a mean average precision (mAP) of 33% on COCO dataset [67]. More generally, one-stage object detectors were mainly introduced to decrease the detection time. These detectors are conceptually different from region proposal networks [68] in that they consider the image as a grid of cells that should be scanned only once.

1) CNN architecture: YOLO has a fully convolutional architecture based on Darknet-53 backbone and an output block within 53 convolutional layers for detection on three different scales. The feature extractor mechanism is based on Features Pyramid Network (FPN) architecture [69] within residual convolutional blocks, and provides feature volume maps as output. YOLO head detector has 1×1 convolutional layers to reduce the depth of the volume activation maps without affecting the spatial resolution. Detection is made in three different scales, which are precisely given by down-sampling the dimensions of the input image by 32, 16 and 8 respectively. The advantage of detecting at three scales is to handle in a better way small object detection. Scales are designed to perform small (map size of 52×52), medium (26×26), and large object detection (13×13). An FPN architecture was implemented within the network to perform the map up-sampling process during detection.

The detector analyzes the input image cells to identify objects and produce outputs: position, size, class confidence score, and class number. The position is given by the center location (C_x, C_y), and the size is determined by the width and height of the bounding box (C_w, C_h). The class confidence score is given for each detected object (C_s). For each bounding box, the model classifies the object as belonging to any of the defined classes C_c. The output vector is in the form $(C_x, C_y, C_w, C_h, C_s, C_c)$ for each bounding box. YOLO has a very deep fully convolutional network based on FPN architecture, which is appropriate for handling scale variations, perspective variations, partial occlusion, and crowded scenes. Except for scale that is not expected to undergo significant change (due to a fixed flight elevation), all the other difficulties are generally frequent in our application context.

2) Training the detector: Training a deep learning object detector from scratch requires a large annotated dataset and significant computational resources. Therefore, we adopt a transfer learning methodology, which consists of retraining a pre-trained model on our dataset. We start the training of our detector, backbone and head detector using weights obtained on a large-scale dataset, such as ImageNet [70], in order to construct a generic mound base-detector.

Our dataset consists of 18 reconstructed orthomosaics, each representing a different planting block. Since the mound annotation task is tedious and time-consuming, we only annotate six orthomosaics, comprising in total of 9661 objects, to be used as a training set for the mound detector. The training set thus represents 33% of the entire orthomosaic dataset, while the remaining 67% are kept for testing. Due to the high resolution of orthomosaics (23610×18151), we adopt a patch-based detection approach, where each orthomosaic is divided into non-overlapped patches using a regular grid with fixed cell size.

3) Data augmentation: Data augmentation has been widely used to overcome the lack of data while training a machine learning model. We apply an augmentation process as a series of modifications on initially annotated object patches, to generate additional examples in the training set. In our application, we aim to train the model to detect the presence of mounds without giving much attention to precise localization. This means that we tolerate detection bounding boxes

Fig. 5: Data augmentation example. Black boxes are the original annotations. Red boxes are examples of generated boxes using the augmentation process.

Algorithm 1: Bounding box augmentation

```
Data: patch P, bounding boxes BB
Result: augmented bounding boxes
for bb ∈ BB do
    transform = random(translation, size);
    if transform = translation then
        Z = random([0,1]);
        Apply equation (1) using Z;
    else if transform = size then
        L = random([0,1]);
        α = random([0,2π]);
        Apply equation (2) using (L, α);
end
```

that are not centered on the object. Therefore, in order to generate new training patches, we apply two modifications to bounding boxes to enhance detection performance with respect to background distraction. The applied modifications are 1) size change and 2) translation. Note that these operations are applied to bounding boxes individually, rather than to the entire orthomosaic. We consider that other transformations, like rotation and blurring, are not necessary, since they are sufficiently present in the dataset. On the other hand, we observed that small-scale changes (caused by differences in sizes between mounds) are efficiently handled by the YOLO architecture described above. Formally, we define our augmentation process by the following operations:

- **Bounding box size change:** we change the scale to include background information from the surrounding region in the bounding box following the equation:

$$\begin{cases} H_{new} = H_{old} \times Z \\ W_{new} = W_{old} \times Z \end{cases} \quad (1)$$

  where H and W are respectively the height and the width of the bounding box, and Z is the scale.
- **Bounding box translation:** we generate non-centered bounding boxes by shifting according to equations:

$$\begin{cases} Cx_{new} = Cx_{old} + (L \times cos(\alpha)) \\ Cy_{new} = Cy_{old} + (L \times sin(\alpha)) \end{cases} \quad (2)$$

  where Cx and Cy are the coordinates of the center of the bounding box, L is the translation amount in pixels and α is the angle defining the translation direction.

The overall augmentation process to generate new patches with augmented bounding boxes is presented in the algorithm (1)

C. Global count estimation

In visual object counting, publicly available datasets are generally fully annotated (e.g. Shanghaitech [45], UCSD [27], UCF_Cc [33], and UCF-QNRF [71]). Therefore, object count generally corresponds to the number of annotated visible objects. However, in our case, the final count is available only after completing planting operations for the corresponding block. Thus, we consider the actual number of planted seedlings as the final count for a given block, which is different from counting manual annotations.

In other words, our objective is to predict the number of tree seedlings to be planted, which is different from the number of visible mounds. This objective cannot be achieved by the only use of the local visual detector due to two main reasons. First, visual detection of mounds is subject to recognition errors under challenging situations, which may result in both missed detection and false positives. Second, the difference between the detection result and the number of plant seedlings finally used may also be explained by the fact that tree seedlings can be planted in positions where there are no visible mounds (e.g. microsite eroded and collapsed by rainwater, completely occluded by woody debris and coarse rock fragments).

In order to predict the number of required tree seedlings for a given block, we define a global count estimator, which consists of an orthomosaic-level model taking as input global information on the planting block. This model is trained from global information on orthomosaics corresponding to planted blocks (i.e. for which the number of plant seedlings finally used is known). Given a set of N training observations $X = \{x^i\}, i = 1, 2...N$ (representing N planting blocks), and corresponding ground-truth count $Y = \{y^i\}, i = 1, 2...N$, for each observation x^i we define a set of M global features $x^i = \{x^i_j\}, j = 1, ..., M$.

We want to learn a mapping function $F : X \rightarrow Y$, to estimate the final count from the set of global features for each block. We use regression analysis as a predictive technique to model the relationship between predictor variables (x^i) and the target variable (y^i). We use ridge regression to minimize the loss function (3) and find the M-dimensional weight vector $W = \{w_j\}, j = 1, ..., M$. The loss function is defined as:

$$\min_{w} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{M} w_j \times x^i_j \right)^2 + \lambda \sum_{j=1}^{M} w_j^2 \quad (3)$$

where y_i is the target variable, x^i_j is the predictor variable, w_j is the weight to be learned, and λ the shrinkage parameter controlling the balance between prediction error and regularization of W.

(a) (b) (c)

Fig. 6: Three patches from different orthomosaics illustrating inter-block variability, which makes mound identification difficult even for humans. (a) The presence of grass causes partial occlusion of mounds. (B) Water accumulation and mound erosion/deterioration are caused by heavy rain events. (c) Dry terrain where mounds have a similar texture to surrounding regions.

To train the regression model, we construct a training dataset where each training example is a global feature vector describing an orthomosaic. Note that prior to constructing the feature vector of an orthomosaic, the local detector is applied in order to obtain a detection-based count as described in section III-D. In addition to detection-based count, the feature vector x^i includes other block-level information, namely, the average detection density per patch, the average mound density according to user input (i.e. fine-tuning annotations), and the planting block area.

D. Processing a new orthomosaic

The variability of terrains in the dataset results in a high appearance variation for both mounds and surrounding areas. Figure 6 shows examples of patches from different orthomosaics, illustrating inter-block variability. To address this issue, we fine-tune the local mound detector before processing a new block, in order to adapt the model to the specificities of the new site (e.g., geographic zone, meteorological conditions, season).

The fine-tuning process aims to increase the detector's ability to discriminate mounds from the surrounding area, in a specific context. In our conception, this is achieved by taking model weights obtained by transfer learning (see section III-B2), and by retraining only the head of the detector, while the backbone layers are fixed. In summary, context adaptation is performed for a given orthomosaic representing a new planting block through the following three steps:

1) Annotating a small batch of patches from the new planting block,
2) Generating a batch of images using data augmentation (see Algorithm 1),
3) Fine-tuning the visual detector using both annotated and generated images.

The fine-tuned detector is then applied to the new orthomosaic to produce a count-by-detection, which is incorporated with global variables to form the block-level feature vector F_i for x^i.

To produce the final count, we finally perform a linear transformation of the feature vector using the estimated weight vector W:

$$Count_{final}(F_i) = \sum_{j=1}^{M} w_j \times x_j^i. \tag{4}$$

Algorithm 2 illustrates the main operations performed when processing a new orthomosaic.

Algorithm 2: processing new orthomosaic

Data: new orthomosaic T, trained detector D, trained global estimator (equation 4)
Result: Final count
begin
  Annotate a small batch of patches ;
  Apply data augmentation (algorithm 1);
  D_f = Fine-tune (D) ;
  C_{det} = Detect (D_f, T) ;
  Extract global features F_{ext} from T ;
  Construct feature vector: $F = (C_{det}, F_{ext})$;
  Final counting: apply equation 4 on F ;
  illustrate
end

IV. Experiments and results

A. Dataset construction

Figure 7 shows different sites in the south of the province of Quebec, Canada, where the study has been conducted. Images were captured at an altitude of 120 m. The sensor was set vertically, and images were captured with a high overlap percentage to maximize orthomosaic reconstruction quality. Orthomosaic reconstruction was performed using the Pix4D software. Terrains included in this work are from different zones and have different characteristics. Figure 3 shows three different orthomosaics after image reconstruction. A total number of 18 orthomosaics have been reconstructed

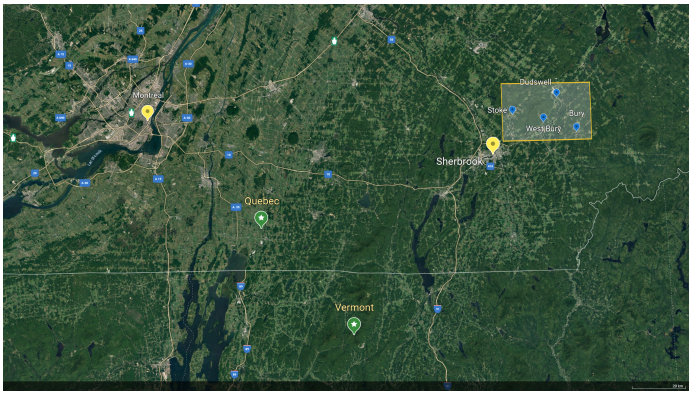


Fig. 7: Geographic zone of the study. The yellow rectangle shows the study area in the south of Quebec, Sherbrook city. Map captured from google earth at an elevation of 244 m.

for 18 planting blocks using the captured images. The number of mounds may vary significantly for different orthomosaics, depending on several factors, such as terrain characteristics and the type of machinery used for mechanical preparation.

We divide our orthomosaic dataset into two groups, G1 and G2.

- **G1:** includes six (6) orthomosaics that we manually annotated for training the object detector and analyzing detection performance. In total, 9661 mounds were manually annotated to train the visual detector. Figure 8 shows manual annotations on examples of extracted patches.
- **G2:** comprises twelve (12) orthomosaics that we mainly use for evaluating prediction performance for the entire framework. Note that mounds are not annotated in this dataset, and only global features are used when processing G2.

To fit the detector input format, each image is divided into a nonoverlapping regular grid, where the cell (patch) size on the grid is 416×416 pixels. This division is performed in a manner to maximize the resolution of image patches (cells), which are processed by the object detector.

Fig. 8: Three annotated patches belonging to different orthomosaics in G1.

B. Evaluation metrics

To evaluate the visual detection performance, we use precision (P), recall (R), average precision (AP), and F1 score defined as follows:

$$P = \frac{TP}{TP + FP} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

$$AP = \sum_{i=0}^{n-1} (R_{i+1} - R_i) P_{interp}(R_{i+1}) \tag{7}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{8}$$

where TP is the number of True Positives, FP is the number of False Positives, and FN is the number of False Negatives. $P_{interp}(R)$ is the precision interpolated at a certain recall level. To evaluate the entire framework performance in the counting task, we adopt the relative counting precision metric defined by:

$$RP = 1 - (\| \frac{\#\text{Predicted\_mounds} - \#\text{GT}}{\#\text{GT}} \|) \tag{9}$$

where #Predicted_mounds is the predicted number of mounds and #GT is the ground truth number.

C. Implementation

The proposed method was implemented using Python on a PC (CPU i7-8700 @ 3.2GHZ, 6 cores) equipped with a GPU Nvidia Geforce GTX 1070. To train YOLOv3, we set the batch size to 16 and the learning rate to 0.001, with a decay of 0.0005. The momentum is set to 0.9 and the number of epochs is fixed to 30 for detector training, and 10 for fine-tuning.

The augmentation parameters Z, L, and α are respectively set to random numbers in $[0.8, 1.2]$, $[1, 10]$, and $[0, 2\pi]$. The regressor parameter λ is set to 10. For constructing the object detector, we set patch size to 256×256 pixels, which produces approximately 10000 training patches after data augmentation, comprising around 95000 annotated mounds for 30 epochs. Transfer learning for YOLO is done based on the weights of the pre-trained model on the ImageNet dataset [70]. For fine-tuning, we use a batch of 11 patches of size 256×256 for each orthomosaic. During the detection process, we set a confidence threshold of 0.25 to detect the most possible mounds without increasing the number of false positives.

Since YOLO implementation was optimized and coded in C, we use the official implementation published by the authors for visual mound detection. We propose a sequential framework to count mounds from orthomosaics. The input is an orthomosaic representing a planting block. First, the input orthomosaic is divided into equal patches of size 608×608 pixels. Then, we annotate a small batch of patches manually to fine-tune the trained visual detector. We fine-tune the model by training the head detector for 10 epochs. After that, the trained model is used to detect mounds in each patch, and results are stored in text files to be used to extract global features. Once visual detection is performed, we construct a global feature vector using annotations and detection results. The total number of the detected mounds is calculated by summing the locally detected mounds on each patch. Then,

TABLE I: Detection accuracy results for the local object detector trained on six different folds.

Fold	Model1				Model2				Model3			
	P	R	AP	F1	P	R	AP	F1	P	R	AP	F1
1	38.4%	16.4%	10.7%	23%	39.3%	19.3%	16.9%	25.9%	30%	21.4%	12.9%	25%
2	44.7%	39.6%	28.3%	42%	41.4%	40.7%	32.5%	41.1%	41.4%	40.7%	32.5%	41.1%
3	61.7%	10.9%	14.7%	18.5%	58.3%	12.1%	16%	20%	23.7%	21.8%	11.1%	0.227
4	34.7%	21.1%	14.3%	26.2%	27.4%	22.9%	14.3%	25%	37.9%	62.5%	40.9%	47.2%
5	35.1%	20.7%	16.8%	26.1%	58.9%	25.1%	24.1%	34.2%	32.4%	36.9%	28.2%	0.345
6	49.2%	29.8%	19.8%	37.1%	35.5%	40.2%	23.5%	37.7%	31.9%	47.5%	28.9%	38.2%
Average	43.96%	23.08%	17.43%	28.82%	43.47%	26.72%	21.22%	30.65%	32.88%	38.47%	**25.75%**	**34.78%**

the average mounds density per patch is calculated for both annotations and detections. Finally, the global feature vector is constructed by adding the planting block area, which is fed to the trained global estimator to predict final count.

The total parameters number of our model is $62,001,757$. The visual object detector inference speed is equal to 45 patches per second. The total time required by our framework to predict the final count depends on the orthomosaic size, which vary between 10000×15000 and 20000×60000 pixels, depending on camera settings and planting block area.

D. Ablation study

1) Local visual detector: In this section, we present an ablation analysis on the $G1$ subset to examine the impact of data augmentation and fine-tuning on detection performance. We evaluate three different scenarios by training and testing three versions of the object detector :

- **Model 1:** We do not use data augmentation so that only annotated data are used for training the detector. For detection, the trained model is applied directly without fine-tuning.
- **Model 2:** We use data augmentation but not fine-tuning.
- **Model 3:** We use the complete version of our detector as described in section III-B2, including data augmentation and fine-tuning steps.

Our evaluation is based on a cross-validation methodology, which performs a circular combination among all possibilities for training and testing. Since the amount of annotated data (6 orthomosaics) is limited compared to the entire dataset (18 orthomosaics), cross-validation is beneficial in that it allows to efficiently exploit available annotations, while mitigating the data split effect [72]–[74].

In each of the six iterations, we partition data as follows.

- Five (5) orthomosaics are used for training/validation. The obtained patches are shuffled and then split into two subsets for training (90%) and validation (10%).
- One (1) orthomosaic is not seen during training and is used only for testing.

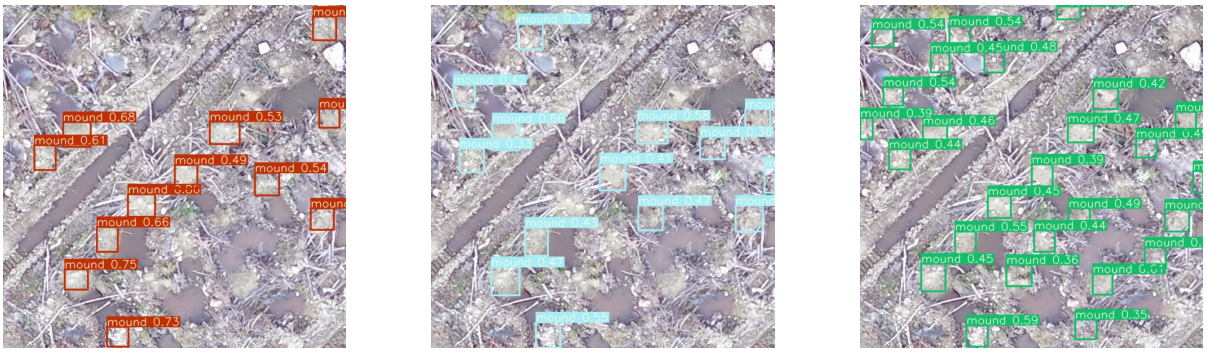
The cross-validation process is illustrated in algorithm 3.

Algorithm 3: K-fold cross validation

Data: K orthomosaics represented by $T = \{T_i\}$, ground-truth counts $Y = \{Y_i\}$
Result: Average error over tests
for $i \in \{1, ..., K\}$ **do**
  Training_set = $\{T \backslash T_i\}$;
  Testing_set = T_i ;
  Detector = Train (Training_set) ;
  h_i = Detect (Detector, Testing_set) ;
  e_i = Calculate_error (h_i, Y_i) ;
end
return $\frac{1}{k} \sum_{i=1}^{k} (e_i)$

Table I shows results of the 6-fold cross-validation for the three detector versions. We report precision (P), recall (R), average precision (AP), and F1 scores for all experiments. Average scores over all iterations (mainly AP and F1) show that training the detector using augmented data enhances recognition performance compared to the first version, where only annotated data are used. This suggests that using only annotated data may cause under-fitting due to the lack of training examples. Thus, the data augmentation process allowed to improve detection performance, which resulted in increasing AP and F1 scores by 3.8% and 1.8%, respectively. However, the best performance over the three scenarios was obtained by using the complete version of the visual detector, which achieved an average detection precision of 25.75% and an F1 score of 34.78%. This result demonstrates the importance of data augmentation during training and fine-tuning when applying the model to a new image.

In Figure 9, we report qualitative results of the three detectors when processing the same image patch of size 498×498, containing 24 objects, according to manual annotations. This example shows that the final version of the detector has a better ability to recognize objects of interest. In fact, augmenting the training set and fine-tuning the detector allowed to increase the detector's ability to discriminate objects from background regions. Moreover, the context information incorporated during fine-tuning is shown to be beneficial in adapting the model to



(a) (b) (c)

Fig. 9: Qualitative results for the three versions of the object detection model on the same patch comprising 24 annotated mounds. The number of detected mounds is 12 for Model 1 (a), 13 for Model 2 (b), and 24 for Model 3 (c).

TABLE II: Relative counting precision of two different versions of the system. Testing is done according to the same cross-validation methodology described in section IV-D1. For each cross-validation iteration, testing is performed on an orthomosaic representing a plantation block.

Orthomosaic	GT	Detection-based count		Globally corrected prediction		
		Count	RP	Count	RP	Improvement
T1	3750	1911	51%	3580	**95.5**%	**45.5**%
T2	1450	1074	74.1%	1166	**80.4**%	**6.3**%
T3	1350	836	**61.9**%	2004	51.6%	-10.3%
T4	4750	5407	86.2%	5051	**93.7**%	**7.5**%
T5	8600	6196	72.0%	8334	**96.9**%	**24.9**%
T6	6500	6866	94.4%	6172	**95**%	**0.6**%
Average			73.2%		**85.4**%	**12.4**%

specific situations.

2) Count correction using global features: To evaluate the contribution of the global correction module in the final prediction, we perform another ablation analysis on the $G1$ dataset. This is done by comparing the counting relative precision of the local visual detector to that of the entire system, including both local detection and global correction process.

Table II presents the counting results of two different versions of our framework. The first version consists of the object detection module used for predicting detection-based count, while the second consists of the entire framework including both detection and correction procedures as illustrated in Figure 5. Results are reported for the $G1$ subset using LOOCV on six different planting blocks. From table II, we can see that the global correction function improved counting precision for five over six planting blocks, with an average improvement of 12.4%. For example, This improvement reached $45,5\%$ for planting block T1. From this experiment, we conclude that counting-by-detection is insufficient to provide precise counting under our application constraints and that global prediction is essential to perform counting correction based on global features of planting blocks.

E. Real-world application results

In real-world conditions, the process of estimating the number of mounds on a new planting block can be summarized as follows:

1) Dividing the input orthomosaic into patches according to a regular grid,
2) Fine-tuning the object detector using a batch of patches,
3) Applying the fine-tuned detector and performing counting-by-detection,
4) Extracting the global feature vector;
5) Predicting the final count using the trained regressor (equation 3).

Note that our training dataset for the local object detector is composed of the six annotated orthomosaics of G1, which are only used for training. Final testing is performed on the remaining 12 orthomosaics of G2 (from T7 to T18). Since we have a limited training set for the regressor (global estimation function), we adopt a leave-one-out cross-validation-like strategy where the regression training set changes for each testing iteration. That is, for the i^{th} test on orthomosaic T_i, the global estimation function is trained on 17 global feature vectors, representing the entire dataset (G1 and G2), except T_i. The main motivation for this choice is to fully take advantage of the limited amount of data for which global feature vectors are available.

TABLE III: Quantitative results of our proposed method. GT is the ground-truth corresponding to the number of plant seedlings planted in a block. The count is the predicted number of mounds and RP is the relative counting precision. We report results for detection-based prediction using YOLO [63] (without global correction), globally corrected prediction (detection followed by correction), and Faster RCNN [68]. The average precision measure represents the average over all precision values, while overall precision stands for the counting precision considering the total number of mounds in the dataset. Note that planting blocks with a larger number of mounds have more significant contributions to the overall precision.

Orthomosaic	GT	YOLO [63]		Faster RCNN [68]		Ours	
		Count	RP	Count	RP	Count	RP
T7	8900	7713	86.7%	4441	50%	8318	**93.5%**
T8	1400	1023	73.1%	1058	76%	1734	**76.1%**
T9	12350	8442	68.4%	5702	46%	11773	**95.3%**
T10	16450	15661	**95.2%**	13467	82%	15226	92.6%
T11	3750	3214	85.7%	1586	42%	4158	**89.1%**
T12	7150	2775	38.8%	3106	43%	6022	**84.2%**
T13	2300	1735	75.4%	803	**35%**	1775	77.2%
T14	6600	1803	27.3%	2607	**40%**	6050	91.7%
T15	4350	2506	57.6%	132	3%	3781	**86.9%**
T16	6150	5829	94%	2354	38%	6019	**97.9%**
T17	2050	771	37.6 %	930	45%	1764	**86.1%**
T18	13100	8914	68%	4572	35%	14364	**90.3%**
Overall result	84550	60386	71.4%	58633	52.8%	80989	**95.8**%
Average precision			67.4%		50.7%		**88.4**%

It can be observed from table III that the comparison between detection-based-counts and globally corrected predictions is consistent with the ablation analysis in IV-D2. This result confirms the importance of combining local visual detection with global count estimation. From detailed results, we can also see that the achieved counting precision is superior to that of the method currently used by forest managers (85%) for nine testing blocks among 12. Moreover, the precision exceeds 90% for six testing blocks, and our framework outperforms the manual method in both average precision (88,4%) and overall precision (95,8%).

We also present a comparison with the state-of-the-art method Faster RCNN [68], which is a two-stage object detector that uses a Region Proposal Network (RPN) to generate proposals. It is clearly seen from table III that our proposed framework achieved better results. Overall, our method improves the performance by 24%, with a 95.8% overall precisioncompared to 52.8% for Faster RCNN. In average precision, which reflects accuracy per block, our method is significantly more effective and achieves 88.4% compared to 67%, and 50% for YOLO and Faster RCNN respectively.

To further validate the importance of our correction module, we conduct a statistical validation student test, which is a statistical hypothesis test used for small set sizes. We claim that the correction module improves the global count and reduces counting error per block. Thus, we define the null H_0 and alternative H_1 hypothesis.

$$\begin{cases} H_0 : \mu_{detection} = \mu_{correction} \\ H_1 : \mu_{detection} \leq \mu_{correction} \end{cases}$$

To calculate the p-value, we use the RP calculated over 12 orthomosaic test data using detection and corrected count. Our calculated p-value is equal to 0.00708 and the statistic value is equal to -3.299. T-test result supports our claim that the correction module improves the final count results. At an alpha value equal to 0.05, the calculated p-value is much lower than the alpha value, which means that the difference between means is statistically significant and the null hypothesis is rejected.

V. Discussion

In this section, we discuss the main challenges related to object detection and final count estimation, based on the obtained results.

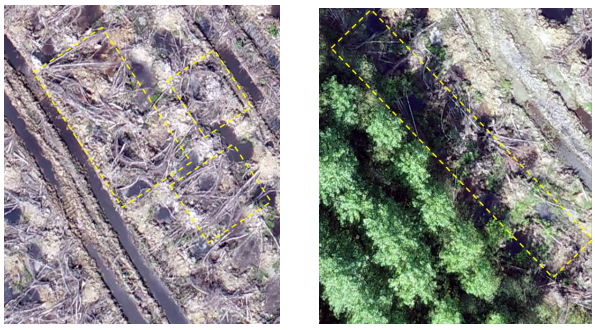
A. Local object detection challenges

The first step in our framework is to infer the orthomosaic to the detector. From table III, we see that for T12 and T14, the counting-by-detection precision is lower than 40%, as the detector was unable to efficiently recognize mounds in corresponding image patches. This can be explained by the relatively limited amount of annotated data available for training the detector (six blocks), compared to the testing set (12 blocks), which may limit the generalization ability of the detection model. In our experiments, an unseen data example is defined as a new plantation block completely excluded from the training set. Therefore, when testing the object detector, it is possible that the new plantation block shows different unseen characteristics. This may include terrain properties and mound shapes that were not encountered by the detector during training.

In order to increase the accuracy of the object detector, the optimal data partitioning strategy would have been to include a sample from each of the 18 blocks in the training set. However, we finally defined a more rigorous evaluation strategy, that is more consistent with our real-world problem and application constraints. To maximize the generalization ability of the model when deploying our application, we recommend training the base detector on all available orthomosaics, or at least including a sample from each available orthomosaic in the training set. We also recommend continuously feeding the system with new data as images and annotations become available, in order to continuously improve detection performance.

Note that even with the use of a larger training dataset, several challenging situations may still significantly impact local detection performance due to recognition perturbation factors. These factors mainly include mound occlusion, destruction, and image acquisition issues.

Occlusion: As illustrated in figure 10.a, this situation may occur due to the presence of woody debris and coarse rock fragments on the forest floor. Moreover, mounds located on block borders may also be occluded by neighboring trees or shadows (see figure 10.(b)).

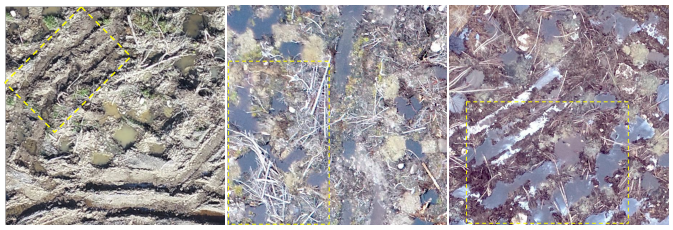


(a) (b)

Fig. 10: Examples of occlusion situations on RGB images captured at an altitude of 120 m. (a) Occlusion is due to the presence of woody debris and rock fragments on the forest floor. (b) Occlusion is due to the presence of trees and shadows on border regions.

Mound destruction: During mechanical preparation of a planting block, mound destruction may be caused by the tracks of the excavator, as shown in figure 11(a). Moreover, in the case of heavy rain events following mechanical preparation, the created mounds may be subject to deterioration and erosion. This is because, in intensive silviculture, planting blocks are often along hillslopes, which favors water flow. Figure 11(b) & (c) shows examples of regions affected by these conditions. In order to handle the effects of hydroclimatic conditions, we recommend performing image capture flights as soon as mechanical preparation is complete.

Image acquisition factors: The image acquisition process is an important step that may significantly impact image quality, as well as the appearance of the object of interest. This is mainly due to several perturbation factors that may occur during flights due to weather conditions, such as camera movements due to the presence of wind and lighting variation. These factors may result in several visual effects on the captured images, like blurred image regions and unbalanced colors (see figure 12). Object visibility is also related to flight altitude, as flying at a low altitude increases the level of detail in images. However, a good trade-off should be found since on the other hand, flying at a high altitude allows to simplify image acquisition and orthomosaic construction, prior to applying the proposed framework.



(a) (b) (c)

Fig. 11: Examples of aerial RGB images showing destructed mounds. (a) Mounds destroyed mounds by the excavator during mechanical preparation. (b) & (c) Mound destruction due to heavy rain events.

(a) (b) (c)

Fig. 12: Examples of visual effects caused by image acquisition conditions. (a) The image is totally blurred. Sunlight reflection increases image brightness in (b), while in (c) the image is relatively dark due to a lack of luminosity.

B. Global prediction performance

The performed experiments show the importance of combining local object detection with a global correction function, in order to improve count estimation performance. This demonstrates that global count estimation based on block-level features (including detection-based count) allows for mitigating the object detection issues discussed above. However, we observed relatively high counting errors during our final tests for plantation blocks T8 and T13 (see table III). We can notice that these two plantation blocks have small areas (2.37 ha and 3.09 ha, respectively), with a small number of mounds (1400 and 2300, respectively). This performance gap for small area blocks can be explained by two reasons. The first is training data imbalance with respect to the area criteria, as small area blocks are underrepresented in the regressor training set. Second, this can also be explained by block border effects,

which are generally more impactful with small blocks (see section V-A and figure 10.b). It would be therefore important to feed the global prediction function with additional global observations of small blocks as soon as they become available, in order to overcome the under-representation issue.

VI. Conclusion

We presented a new computer vision framework to automate the counting process of planting microsites on a mechanically prepared planting block. The proposed system is based on a hybrid approach combining local patch-level detection with global prediction at the block level. The performed experiments demonstrate the effectiveness of our design in handling several challenging situations related to environmental and image acquisition conditions, as well as to inherent limitations of the detection model. Indeed, while the object detection model exploits visual information from UAV images to predict a preliminary count, the subsequent global estimation procedure exploits complementary block-level information to mitigate detection limitations. Furthermore, the proposed solution outperforms traditional counting methods, while offering significant advantages in terms of fieldwork conditions and cost optimization.

Our future work comprises two main lines of research. At the fundamental level, we aim to incorporate both local detection and global correction mechanisms in a single end-to-end deep learning model to address the counting problem. At the application level, we aim at generalizing the model to other computer vision applications implying object counting in crowded scenes.

References

[1] J. Wolken, S. Landhäusser, V. Lieffers, and M. Dyck, "Differences in initial root development and soil conditions affect establishment of trembling aspen and balsam poplar seedlings," *Botany*, vol. 88, pp. 275–285, 2010.

[2] A. A. D. Santos, J. M. Junior, M. S. Araújo, D. R. D. Martini, E. C. Tetila, H. L. Siqueira, C. Aoki, A. Eltner, E. Matsubara, H. Pistori, R. Feitosa, V. Liesenberg, and W. N. Gonçalves, "Assessment of cnn-based methods for individual tree detection on images captured by rgb cameras attached to uavs," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[3] P. Chamoso, W. Raveane, V. Parra, and M. A. G. Arrieta, "Uavs applied to the counting and monitoring of animals," in *ISAmI*, 2014.

[4] S. Baena, J. Moat, O. Q. Whaley, and D. Boyd, "Identifying species from the air: Uavs and the very high resolution challenge for plant conservation," *PLoS ONE*, vol. 12, 2017.

[5] S. F. D. Gennaro, C. Nati, R. Dainelli, L. Pastonchi, A. Berton, P. Toscano, and A. Matese, "An automatic uav based segmentation approach for pruning biomass estimation in irregularly spaced chestnut orchards," *Forests*, vol. 11, p. 308, 2020.

[6] J. il Shin, W. woo Seo, T. Kim, J. Park, and C. shik Woo, "Using uav multispectral images for classification of forest burn severity—a case study of the 2019 gangneung forest fire," *Forests*, vol. 10, p. 1025, 2019.

[7] S. Velastin, J. Yin, A. Davies, M. Vicencio-Silva, R. Allsop, and A. Penn, "Analysis of crowd movements and densities in built-up environments using image processing," in *IEE Colloquium on Image Processing for Transport Applications*, 1993, pp. 8/1–8/6.

[8] C. Regazzoni, A. Tesei, and V. Murino, "A real-time vision system for crowding monitoring," *Proceedings of IECON '93 - 19th Annual Conference of IEEE Industrial Electronics*, pp. 1860–1864 vol.3, 1993.

[9] A. C. Davies, J. Yin, and S. Velastin, "Crowd monitoring using image processing," *Electronics & Communication Engineering Journal*, vol. 7, pp. 37–47, 1995.

[10] C. Regazzoni and A. Tesei, "Distributed data fusion for real-time crowding estimation," *Signal Process.*, vol. 53, pp. 47–63, 1996.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[12] P. A. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004.

[13] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, pp. 247–266, 2006.

[14] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[15] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 878–885 vol. 1, 2005.

[16] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1713–1727, 2008.

[17] M. Enzweiler and D. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 2179–2195, 2009.

[18] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, 2009.

[19] S.-F. Lin, J.-Y. Chen, and H.-X. Chao, "Estimation of number of people in crowded scenes using perspective transformation," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 31, pp. 645–654, 2001.

[20] T. Zhao, R. Nevatia, and B. Wu, "Segmentation and tracking of multiple humans in crowded environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1198–1211, 2008.

[21] J. Ilao and M. Cordel, "Crowd estimation using region-specific hog with svm," *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 1–5, 2018.

[22] B. Zhou, M. Lu, and Y. Wang, "Counting people using gradient boosted trees," *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pp. 391–395, 2016.

[23] V. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, "Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3253–3261, 2015.

[24] A. B. Chan and N. Vasconcelos, "Bayesian poisson regression for crowd counting," *2009 IEEE 12th International Conference on Computer Vision*, pp. 545–551, 2009.

[25] D. Ryan, S. Denman, C. Fookes, and S. Sridharan, "Crowd counting using multiple local features," *2009 Digital Image Computing: Techniques and Applications*, pp. 81–88, 2009.

[26] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting," in *BMVC*, 2012.

[27] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7, 2008.

[28] K. Mikolajczyk, A. Zisserman, and C. Schmid, "Shape recognition with edge-based features," in *BMVC*, 2003.

[29] M. Tüceryan and A. K. Jain, "Texture analysis," in *Handbook of Pattern Recognition and Computer Vision*, 1993.

[30] J. Hwang and H.-S. Lee, "Adaptive image interpolation based on local gradient features," *IEEE Signal Process. Lett.*, vol. 11, pp. 359–362, 2004.

[31] N. Paragios and V. Ramesh, "A mrf-based approach for real-time subway monitoring," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[32] A. Marana, L. F. Costa, R. Lotufo, and S. Velastin, "On the efficacy of texture analysis for crowd monitoring," *Proceedings SIBGRAPI'98. International Symposium on Computer Graphics, Image Processing, and Vision (Cat. No.98EX237)*, pp. 354–361, 1998.

[33] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2547–2554, 2013.

[34] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *NIPS*, 2010.

[35] Y. Wang and Y. Zou, "Fast visual object counting via example-based density estimation," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3653–3657, 2016.

[36] B. Xu and G. Qiu, "Crowd density estimation based on rich features and random projection forest," *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8, 2016.

[37] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep people counting in extremely dense crowds," *Proceedings of the 23rd ACM international conference on Multimedia*, 2015.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[39] M. Fu, P. Xu, X. Li, Q. Liu, M. Ye, and C. Zhu, "Fast crowd density estimation with convolutional neural networks," *Eng. Appl. Artif. Intell.*, vol. 43, pp. 81–88, 2015.

[40] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 833–841, 2015.

[41] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *ECCV*, 2016.

[42] C. Shang, H. Ai, and B. Bai, "End-to-end crowd counting via joint learning local and global count," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1215–1219, 2016.

[43] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3642–3649, 2012.

[44] L. Boominathan, S. Kruthiventi, and R. V. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," *Proceedings of the 24th ACM international conference on Multimedia*, 2016.

[45] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 589–597, 2016.

[46] D. Oñoro-Rubio and R. López-Sastre, "Towards perspective-free object counting with deep learning," in *ECCV*, 2016.

[47] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4031–4039, 2017.

[48] V. Ranjan, H. M. Le, and M. Hoai, "Iterative crowd counting," *ArXiv*, vol. abs/1807.09959, 2018.

[49] L. Zhang, M. Shi, and Q. Chen, "Crowd counting via scale-adaptive convolutional neural network," *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1113–1121, 2018.

[50] X. Jiang, Z. Xiao, B. Zhang, X. Zhen, X. Cao, D. Doermann, and L. Shao, "Crowd counting and density estimation by trellis encoder-decoder networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6126–6135, 2019.

[51] L. Dong, H. Zhang, Y. Ji, and Y. Ding, "Crowd counting by using multi-level density-based spatial information: A multi-scale cnn framework," *Inf. Sci.*, vol. 528, pp. 79–91, 2020.

[52] Y. Hu, X. Jiang, X. Liu, B. Zhang, J. Han, X. Cao, and D. S. Doermann, "Nas-count: Counting-by-density with neural architecture search," in *ECCV*, 2020.

[53] Y. Liu, G. Cao, H. Shi, and Y. Hu, "Lw-count: An effective lightweight encoding-decoding crowd counting network," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2022.

[54] S. Aldhaheri, R. M. Alotaibi, B. A. M. Alzahrani, A. Hadi, A. Mahmood, A. M. Alhothali, and A. Barnawi, "Macc net: Multi-task attention crowd counting network," *Applied Intelligence*, 2022.

[55] B. Sheng, C. Shen, G. Lin, J. Li, W. Yang, and C. Sun, "Crowd counting via weighted vlad on a dense attribute feature map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, pp. 1788–1797, 2018.

[56] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3304–3311, 2010.

[57] N. Ilyas, A. Ahmad, and K. Kim, "Casa-crowd: A context-aware scale aggregation cnn-based crowd counting technique," *IEEE Access*, vol. 7, pp. 182 050–182 059, 2019.

[58] Y. Tian, Y. Lei, J. Zhang, and J. Z. Wang, "Padnet: Pan-density crowd counting," *IEEE Transactions on Image Processing*, vol. 29, pp. 2714–2727, 2020.

[59] Y. Lei, Y. Liu, P. Zhang, and L. Liu, "Towards using count-level weak supervision for crowd counting," *Pattern Recognit.*, vol. 109, p. 107616, 2021.

[60] P. T. Do, "Attention in crowd counting using the transformer and density map to improve counting result," in *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, 2021, pp. 65–70.

[61] W. Liu, J. Zhou, B. Wang, M. Costa, S. M. Kaeppler, and Z. Zhang, "Integratenet: A deep learning network for maize stand counting from uav imagery by integrating density and local count maps," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[62] Y.-K. Lin, C.-F. Wang, C.-Y. Chang, and H. Sun, "An efficient framework for counting pedestrians crossing a line using low-cost devices: the benefits of distilling the knowledge in a neural network," *Multim. Tools Appl.*, vol. 80, pp. 4037–4051, 2021.

[63] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[64] W. Bouachir, K. E. Ihou, H. Gueziri, N. Bouguila, and N. Bélanger, "Computer vision system for automatic counting of planting microsites using uav imagery," *IEEE Access*, vol. 7, pp. 82 491–82 500, 2019.

[65] G. Hamed, M. A. E.-R. Marey, S. A. El-Sayed, and M. F. Tolba, "Yolo based breast masses detection and classification in full-field digital mammograms," *Computer methods and programs in biomedicine*, p. 105823, 2021.

[66] W.-Y. Hsu and W.-Y. Lin, "Ratio-and-scale-aware yolo for pedestrian detection," *IEEE Transactions on Image Processing*, vol. 30, pp. 934–947, 2021.

[67] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[68] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[69] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[70] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

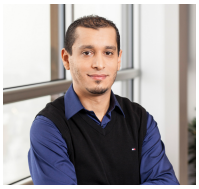
[71] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, and M. Shah, "Composition loss for counting, density map estimation and localization in dense crowds," *ArXiv*, vol. abs/1808.01050, 2018.

[72] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society series b-methodological*, vol. 36, pp. 111–133, 1974.

[73] T. Fushiki, "Estimation of prediction error by using k-fold cross-validation," *Statistics and Computing*, vol. 21, pp. 137–146, 2011.

[74] A. Kopper, R. Karkare, R. Paffenroth, and D. Apelian, "Model selection and evaluation for machine learning: Deep learning in materials processing," *Integrating Materials and Manufacturing Innovation*, vol. 9, pp. 287 – 300, 2020.

Ahmed Zgaren is a Ph.D. student in the Department of Information Systems and Engineering at Concordia University (Montréal, Canada). He holds the Master degree in computer science from the National Engineering School of Tunis (Tunisia), and the engineer degree from the Military College of Tunisia. His research interests include computer vision, object detection/tracking, UAV imaging, and speech analysis.



Wassim Bouachir is currently a professor of computer science at the University of Québec (TÉLUQ). He holds a Ph.D. degree in computer engineering from Polytechnique Montréal and a M.Sc. in computer science from the University of Moncton. His research interests include fundamental problems in computer vision, signal processing, and machine learning. His research activities also aim to develop AI-based systems for several application areas, such as security, physical and mental health, and environment applications.

Nizar Bouguila (Senior member IEEE) received the engineer degree from the University of Tunis, Tunis, Tunisia, in 2000, and the M.Sc. and Ph.D. degrees in computer science from Sherbrooke University, Sherbrooke, QC, Canada, in 2002 and 2006, respectively. He is currently a Professor with the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University, Montreal, Quebec, Canada. His research interests include image processing, machine learning, data mining,, computer vision, and pattern recognition. Prof. Bouguila received the best Ph.D Thesis Award in Engineering and Natural Sciences from Sherbrooke University in 2007. He was awarded the prestigious Prix d'excellence de l'association des doyens des etudes superieures au Quebec (best Ph.D Thesis Award in Engineering and Natural Sciences in Quebec), and was a runner-up for the prestigious NSERC doctoral prize. He was the holder of a Concordia University research Chair Tier 2 from 2014 to 2019 and was named Concordia University research Fellow in 2020. He is the author or co-author of more than 400 publications in several prestigious journals and conferences. He is a regular reviewer for many international journals and serving as associate editor for several journals such as Pattern Recognition journal. Dr. Bouguila is a licensed Professional Engineer registered in Ontario, and a Senior Member of the IEEE.