Multispecies Fruit Flower Detection Using a Refined Semantic Segmentation Network

Philipe A. Dias, Amy Tabb, and Henry Medeiros

***Abstract*—In fruit production, critical crop management decisions are guided by bloom intensity, i.e., the number of flowers present in an orchard. Despite its importance, bloom intensity is still typically estimated by means of human visual inspection. Existing automated computer vision systems for flower identification are based on hand-engineered techniques that work only under specific conditions and with limited performance. This letter proposes an automated technique for flower identification that is robust to uncontrolled environments and applicable to different flower species. Our method relies on an end-to-end residual convolutional neural network (CNN) that represents the state-of-the-art in semantic segmentation. To enhance its sensitivity to flowers, we fine-tune this network using a single dataset of apple flower images. Since CNNs tend to produce coarse segmentations, we employ a refinement method to better distinguish between individual flower instances. Without any preprocessing or dataset-specific training, experimental results on images of apple, peach, and pear flowers, acquired under different conditions demonstrate the robustness and broad applicability of our method.**

***Index Terms*—Bloom intensity estimation, flower detection, semantic segmentation networks, precision agriculture.**

I. Introduction

BLOOM intensity corresponds to the number of flowers present in orchards during the early growing season. Climate and bloom intensity information are crucial to guide the processes of pruning and thinning, which directly impact fruit load, size, coloration, and taste [1], [2]. Accurate estimates of bloom intensity can also benefit packing houses, since early crop-load estimation greatly contributes to optimizing postharvest handling and storage processes.

Visual inspection is still the dominant approach for bloom intensity estimation in orchards, a technique which is time-consuming, labor-intensive and prone to errors [3]. Since only a limited sample of trees is inspected, the extrapolation to the entire orchard relies heavily on the grower's experience. Moreover, it does not provide information about the spatial variability in the orchard, although the benefits of precision agriculture practices are well known [4].

These limitations added to the short-term nature of flower appearance until petal fall make an automated method highly desirable. Multiple automated computer vision systems have been proposed to solve this problem, but most of these methods rely on hand-engineered features [5], making their overall performance acceptable only under relatively controlled environments (e.g., at night with artificial illumination). Their applicability is in most cases species-specific and highly vulnerable to variations in lightning conditions, occlusions by leaves, stems or other flowers [6].

In the last decade, deep learning approaches based on convolutional neural networks (CNNs) led to substantial improvements in the state-of-the-art of many computer vision tasks [7]. Recent works have adapted CNN architectures to agricultural applications such as fruit quantification [8], classification of crops [9], and plant identification from leaf vein patterns [10]. To the best of our knowledge, our work in [11] was the first to employ CNNs for flower detection. In that work, we combined superpixel-based region proposals with a classification network to detect apple flowers. Limitations of that approach are intrinsic to the inaccuracies of superpixel segmentation and the network architecture.

In the present work, we provide the following contributions for automated flower segmentation:

- A novel technique for flower identification that is i) automated, ii) robust to clutter and changes in illumination; and, iii) generalizable to multiple species. Using as starting point a fully convolutional network (FCN) [12] pre-trained on a large multi-class dataset, we describe an effective fine-tuning procedure that adapts this model for fine pixel-wise flower segmentation. Our final method evaluates in less than 50 seconds high-resolution images covering each a full tree. Although the task comparison is not one-to-one, human workers may need on average up to 50 minutes to count the number of flowers per tree.
- A feasible procedure for evaluating high-resolution images with deep FCNs on commercial GPUs. Fully convolutional computations require GPU memory space that exponentially increases according to image resolution. We employ an image partitioning mechanism with partially overlapping windows, which reduces artifacts introduced by artificial boundaries when evaluating disjoint image regions.
- Release of an annotated dataset with pixel-accurate labels for flower segmentation on high resolution images [13].

Manuscript received February 24, 2018; accepted June 10, 2018. Date of publication June 22, 2018; date of current version July 5, 2018. This letter was recommended for publication by Associate Editor I. Sa and Editor C. Stachniss upon evaluation of the reviewers comments. This work was supported by USDA-ARS Agreement #584080-5-020. This paper is part of IEEE Robotics and Automation Letters' Special Issue on Precision Agricultural Robotics and Autonomous Farming Technologies, edited by H. S. Ahn, I. Sa, and F. Dayoub. *(Corresponding author: Philipe A. Dias.)*

P. A. Dias and H. Medeiros are with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI 53233 USA (e-mail: philipe.ambroziodias@marquette.edu; henry.medeiros@marquette.edu).

A. Tabb is with the U.S. Department of Agriculture, Kearneysville, WV 25430 USA (e-mail: amy.tabb@ars.usda.gov).

Digital Object Identifier 10.1109/LRA.2018.2849498

We believe this can greatly benefit the community, since this is a very time consuming yet critical task for both training and evaluation of segmentation models.

II. Related Work

Previous attempts at automating bloom intensity estimation were mostly based on color thresholding, such as the works described in [14], [15] and [16]. Despite differences in terms of color-space used for analysis (e.g., HSL and RGB), all these methods fail when applied in uncontrolled environments. Apart from size filtering, no morphological feature is taken into account, such that thresholding parameters have to be adjusted in case of changes in illumination, camera position or flowering density. Even strategies using aerial multispectral images such as [17] also rely solely on color information for image processing.

Our previous work in [11] introduced a novel approach for apple flower detection that relies on a fine-tuned Clarifai CNN [18] to classify individual superpixels composing an image. That method highly outperformed color-based approaches, especially in terms of generalization to datasets composed of different flower species and acquired in uncontrolled environments. However, existing superpixel algorithms rely solely on local context information, representing the main source of imprecisions in scenarios where flowers and the surrounding background present similar colors.

While early attempts for autonomous fruit detection also relied on hand-engineered features (e.g., color, texture, shape) [6], recent works have been exploring more advanced computer vision techniques. One example is the work of Hung *et al.* [19], which combines sparse autoencoders [7] and support vector machines (SVM) for segmenting leaves, almonds, trunks, ground and sky. The approaches described by Bargoti and Underwood in [20] and Chen *et al.* in [8] for fruit detection share some similarities with our method for flower segmentation. In [20], the authors introduce a Faster R-CNN trained for the detection of mangoes, almonds and apple fruits on trees. The method introduced in [8] for counting apples and oranges employs a fully convolutional network (FCN) to perform fruit segmentation and a convolutional network to estimate fruit count.

End-to-end fully convolutional networks [21] have been replacing traditional fully connected architectures for image segmentation tasks [22]. Conventional architectures such as the *Alexnet* [23] and VGG [24] networks are very effective for image classification but provide coarse outputs for image segmentation tasks. This is a consequence of the image downsampling introduced by the *max-pooling* and *striding* operations performed by these networks, which allow the extraction of learned hierarchical features at the cost of pixel-level precision [12].

Different strategies have been proposed to alleviate the effects of downsampling [22], including the use of deconvolution layers [21], [25], and encoder-decoder architectures with skip layer connections [26], [27]. The DeepLab model introduced in [12] is one of the most successful approaches for semantic image segmentation using deep learning. By combining the ResNet-101 [28] model with *atrous* convolutions and spatial pyramid pooling, it significantly reduces the downsampling rate and achieves state-of-the-art performance in challenging semantic segmentation datasets such as the PASCAL VOC [29] and COCO [30].

In addition to the changes in CNN architecture, the authors of DeepLab also employ the dense CRF model described in [31] to produce fine-grained segmentations. Although providing visually appealing segmentations, this refinement model relies on parameters that have to be optimized by means of supervised grid-search. In [32], we introduced a generic post-processing module that can be coupled to the output of any CNN to refine segmentations without the need for dataset-specific tuning. Called region growing refinement (RGR), this algorithm uses the score maps available from the CNN to divide the image into regions of high confidence background, high confidence object and uncertainty region. By means of appearance-based region growing, pixels within the uncertainty region are classified based on initial seeds randomly sampled from the high confidence regions.

III. Our Approach

In this section, we first describe the pre-training and fine-tuning procedures carried out to obtain a CNN highly sensitive to flowers. Subsequently, we describe the sequence of operations that our pipeline performs to segment flowers in an image.

A. Network Training

One of the largest datasets available for semantic segmentation, the COCO dataset [30] was recently augmented by Caesar *et al.* [33] into the COCO-Stuff dataset. This dataset includes pixel-level annotations of classes such as *grass, leaves, tree* and *flowers*, which are relevant for our application. In the same work, the authors also discuss the performance of modern semantic segmentation methods on COCO-Stuff, with a DeepLab-based model outperforming the standard FCN. Thus, we opted for the publicly available DeepLab-ResNet model pre-trained on the COCO-Stuff dataset as the starting point for our pipeline. Rather than fine-tuning the dense CRF model used in the original DeepLab work, we opt for the generic RGR algorithm as a post-processing module to obtain fine-grained segmentations.

The base model was originally designed for segmentation within the 172 COCO-Stuff classes. To adapt its architecture for our binary flower segmentation task, we perform procedures known as *network surgery* and *fine-tuning* [34]. The surgery procedure is analogous to the pruning of undesired branches in trees: out of the original 172 classification branches, we preserve only the weights and connections responsible for the segmentation of classes of interest.

We considered first an architecture preserving only the *flower* classification branch, followed by a sigmoid classification unit. However, without the normalization induced by the model's original softmax layer, the scores generated by the transferred *flower* branch are unbounded and the final sigmoid easily saturates. To alleviate the learning difficulties caused by such a poor initialization, we opted for tuning a model with two-branches, under the hypothesis that a second branch would allow the

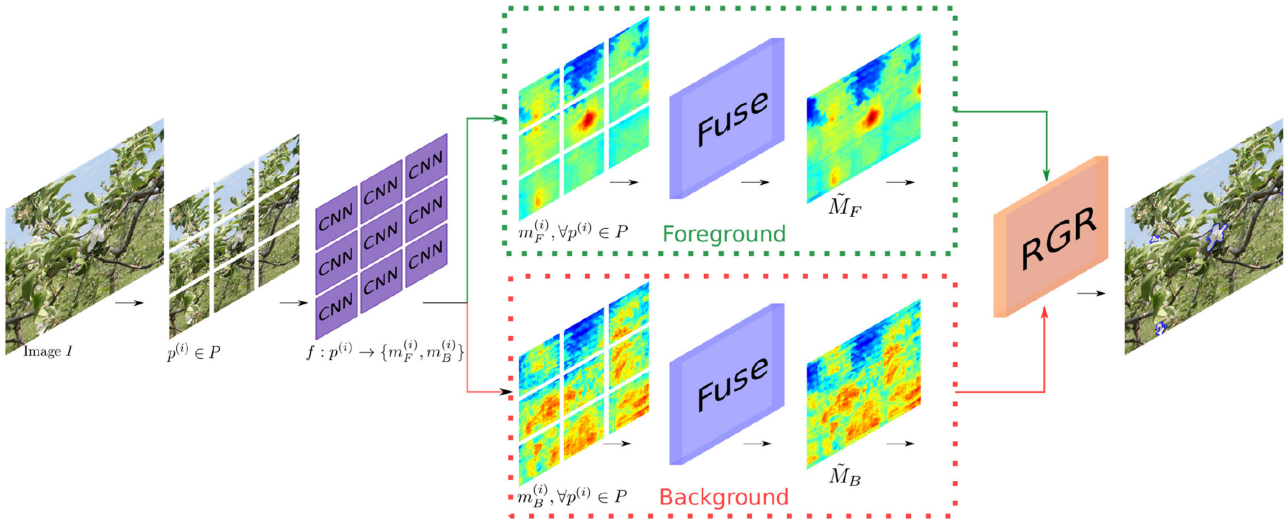

Fig. 1. **Best viewed in color.** Diagram illustrating the sequence of tasks performed by the proposed method for flower detection. Each task and its corresponding output (shown below the arrows) are described in Algorithm 1. In the heatmaps, blue is associated with lower scores, while higher scores are illustrated with red.

network to learn a background representation that properly normalizes the predictions generated by the foreground (*flower*) branch.

We have observed experimentally that nearby leaves represent one of the main sources of misclassification for flower segmentation. Moreover, predictions for the class *leaf* presented the highest activations when applying the pre-trained model to our training dataset. For these reasons, we opt for this branch together with the one associated with *flowers* to initialize our two-branch flower segmentation network.

The adapted architecture was then fine-tuned using the training set described in Section IV, which contains 100 images of apple trees. For our experiments, the procedure was carried out for 10,000 iterations using the Caffe framework [35], with an initial learning rate of 10^{-4} that polynomially decays according to $10^{-4} \times (1 - i/10000)^{0.9}$, where i is the iteration number. Aiming at scale robustness, our fine-tuning procedure employs the same strategy used for model pre-training, where each training portrait is evaluated at (0.5, 0.75, 1.0, 1.25, 1.5) times its original resolution.

While the validation set has pixel-accurate annotations obtained using the procedure described in Section IV, the training set was annotated using the less precise but quicker superpixel-based procedure described in our previous work [11]. Less than 5% of the total image areas in this dataset contain flowers. To compensate for this imbalance, we augmented portraits containing flowers by mirroring them with respect to vertical and horizontal axes. Following the original network parameterization, we split the 100 training images into portraits of 321×321 pixels, corresponding to a total of 52,644 training portraits after augmentation.

B. Segmentation Pipeline

The method we propose for fruit flower segmentation consists of three main operations: 1) divide a high resolution image into smaller patches, in a sliding window manner; 2) evaluate each patch using our fine-tuned CNN; 3) apply the refinement algorithm on the obtained scoremaps to compute the final segmentation mask. These steps are described in detail below. In our description, we make reference to Algorithm 1 and Fig. 1.

Algorithm 1: Proposed approach for flower detection.

Input: Image I.
Output: Estimated flower segmentation map $\hat{Y}$ of image I.
1: Sliding window: divide I into a set of n portraits P.
2: **for** each portrait $p^{(i)} \in P$ **do**
3: Compute scoremaps $m_B^{(i)}$ and $m_F^{(i)}$ using the fine-tuned CNN
4: **end for**
5: Obtain M_B and M_F by fusing $m_B^{(i)}$ and $m_F^{(i)}$ $(i = 1, \ldots, n)$, respectively according to Eq. 2.
6: Normalize M_B and M_F into $\tilde{M}_B$ and $\tilde{M}_F$, respectively according to Eq. 3.
7: Generate $\hat{Y}$ by applying RGR to $\tilde{M}_B$ and $\tilde{M}_F$.

1) Step 1 - Sliding window: As mentioned above, the adopted CNN architecture either crops or resizes input images to 321×321 portraits. Since our datasets are composed of images with resolution ranging from 2704×1520 to 5184×3456 pixels (see Section IV), we emulate a sliding window approach to avoid resampling artifacts. More specifically, we split each input image I into a set P of n portraits $p^{(i)} \in P$. Each portrait is 321×321 pixels large, i.e., $p^{(i)} \in \mathbb{R}^{r \times r}$ with $r = 321$. Cropping non-overlapping portraits from the original image introduces artificial boundaries that compromise the detection quality. For this reason, in our approach each portrait overlaps a percentage s of the area of each immediate neighbor. For our experiments, we adopted $s = 10\%$. When the scoremaps are fused, the results corresponding to the overlapping pixels are discarded. Fig. 2 illustrates this process for a pair of subsequent portraits. The scores obtained for each portrait are depicted as a

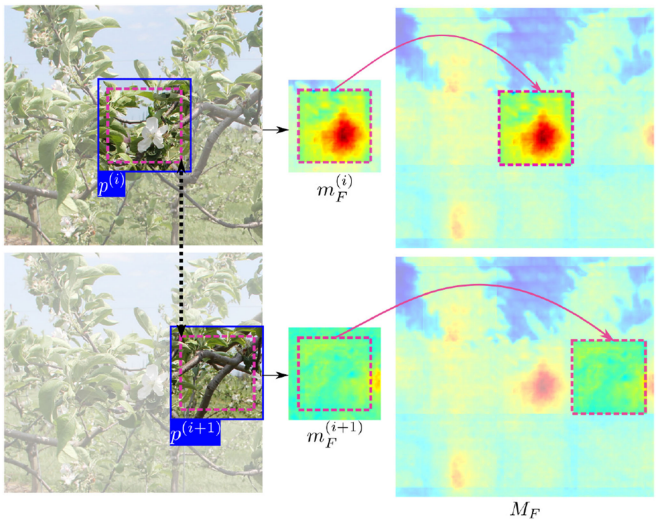

Fig. 2. **Best viewed in color.** Illustration of the sliding window and subsequent fusion process that comprise our segmentation pipeline. Each portrait overlaps a certain area of its neighbors, which is discarded during fusion to avoid artifacts caused by artificial boundaries.

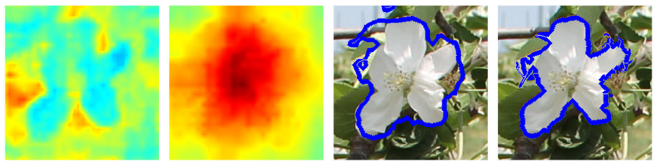

Fig. 3. **Best viewed in color.** Example of segmentation refinement for a given pair of scoremaps. (a) Background scoremap $m_B^{(i)}$. (b) Foreground scoremap $m_F^{(i)}$. (c) Coarse segmentation by direct thresholding of the scoremaps. (d) Refined segmentation using RGR.

heatmap, where blue is associated with lower scores and higher scores are illustrated with red.

2) Step 2 - CNN prediction: We evaluate in parallel each portrait $p^{(i)}$ with our fine-tuned network for flower identification. The CNN is equivalent to a function f

$$f : p^{(i)} \rightarrow \{m_F^{(i)}, m_B^{(i)}\}, \tag{1}$$

which maps each input $p^{(i)}$ into two pixel-dense scoremaps: $m_F^{(i)} \in \mathbb{R}^{r \times r}$ represents the pixel-wise likelihood that pixels in $p^{(i)}$ belong to the foreground (i.e., flower), while $m_B^{(i)} \in \mathbb{R}^{r \times r}$ corresponds to the pixel-wise background likelihood. The heatmaps in Figures 3(a) and (b) are examples of scoremaps computed for a given portrait.

3) Step 3 - Fusion and refinement: After evaluating each portrait, we generate two global scoremaps M_B and M_F by combining the predictions obtained for all $p^{(i)} \in P$. Let $c^{(i)}$ represent the pixel-coordinates of $p^{(i)}$ in I after discarding the padding pixels. The fusion procedure is defined as

$$\forall p^{(i)} \in P, \quad M_{F,B}(c^{(i)}) = m_{F,B}^{(i)}, \tag{2}$$

such that both scoremaps M_B and M_F have the same resolution as I. As illustrated in Fig. 2, the padded areas of $m_{F,B}^{(i)}$ (outside the red box) are discarded during fusion. For every pixel in the image, a single prediction score is obtained from exactly one portrait, such that artifacts introduced by artificial boundaries are avoided.

After fusion, the scoremaps M_B and M_F are normalized into scoremaps $\tilde{M}_B$ and $\tilde{M}_F$ using a softmax function

$$\tilde{M}_{F,B}(q_j) = \frac{\exp(M_{F,B}(q_j))}{\exp(M_B(q_j)) + \exp(M_F(q_j))} \tag{3}$$

where q_j is the j-th pixel in the input image I. With this formulation, for each pixel q_j the scores $\tilde{M}_B(q_j)$ and $\tilde{M}_F(q_j)$ add to one, i.e., they correspond to the probability that q_j belongs to the corresponding class.

As Fig. 3(c) shows, the predictions obtained directly from the CNN are coarse in terms of adherence to actual flower boundaries. Therefore, rather than directly thresholding $\tilde{M}_F$, this scoremap and the image I are fed to the RGR refinement module described in [32]. For our application, the refinement algorithm relies on two high-confidence classification regions R_F and R_B defined according to

$$R_{F,B} = \left\{ q_j | \tilde{M}_{F,B}(q_j) > \tau_{F,B} \right\} \tag{4}$$

where τ_B and τ_F are the high-confidence background and foreground thresholds. Using the high-confidence regions as starting points, the RGR algorithm performs multiple Monte Carlo region growing steps that groups similar pixels into clusters. Afterwards, it performs majority voting to classify each cluster according to the presence of flowers. Each pixel q_j within a cluster contributes with a positive vote if its score $\tilde{M}_F(q_j)$ is larger than a threshold τ_0. As detailed in Section V, this parameter can be empirically tuned according to the dataset under consideration. Based on a grid-search optimization on our training dataset, we selected $\tau_0 = 0.3$ for all our experiments and fixed $\tau_B = 0.1$ and $\tau_F = 1.25 \times \tau_0$.

IV. Datasets

We evaluate our method on four datasets that we created and made publicly available: *AppleA*, *AppleB*, *Peach*, *Pear* [13]. As summarized in Table I, images from different fruit flower species were collected in diverse uncontrolled environments and under different angles of capture.

Both datasets *AppleA* and *AppleB* are composed of images of apple trees, which were collected in a USDA orchard on a sunny day. In both datasets, the trees are supported with trellises and planted in rows. *AppleA* is a collection of 147 images acquired using a hand-held camera. From this total, we randomly selected 100 images to build the training set used to train the CNN. Out of the remaining 47 images, 30 were randomly selected to compose the testing set for which we report results in Section V.

This dataset contains flowers that greatly vary in terms of size, cluttering, occlusion by leaves and branches. Flowers composing its images have an average area of 10,730 pixels, but with a standard deviation of 17,150 pixels. On average, flowers compose only 2.5% of the total image area within this dataset, which is otherwise vastly occupied by leaves.

TABLE I
DATASETS SPECIFICATIONS

Dataset	No. images	Weather	Background panel	Camera model	Resolution	Camera support
AppleA	100 (train) + 30 (val)	Sunny	No	Canon EOS 60D	5184×3456	Hand-held
AppleB	18	Sunny	Yes	GoPro HERO5	2704×1520	Utility vehicle
Peach	24	Overcast	No	GoPro HERO5	2704×1520	Hand-held
Pear	18	Overcast	No	GoPro HERO5	2704×1520	Hand-held

Fig. 4. **Best viewed in color.** Utility vehicle used for imaging. For the *AppleB* dataset, this vehicle was used in conjunction with a background panel.

Fig. 5. **Best viewed in color.** Example of ground truth obtained from freehand annotations. *Left:* positive examples are annotated in blue, while hard negatives are indicated in red. *Right:* segmentation obtained after RGR refinement.

Differently from *AppleA*, for the *AppleB* dataset, a utility vehicle equipped with a background unit was used for imaging, such that trees in other rows are not visible in the images. Fig. 4 illustrates the utility vehicle used for image acquisition, and Figures 6 and 7 illustrate the differences between datasets *AppleA* and *AppleB*.

The *Peach* and *Pear* datasets differ both in terms of species and acquisition conditions, therefore representing adequate scenarios for evaluating the generalization capabilities of the proposed method. Both datasets contain images acquired on an overcast day and without a background unit. Compared to the *AppleA* dataset, images composing these datasets present significantly lower saturation and value means. Tables II and III

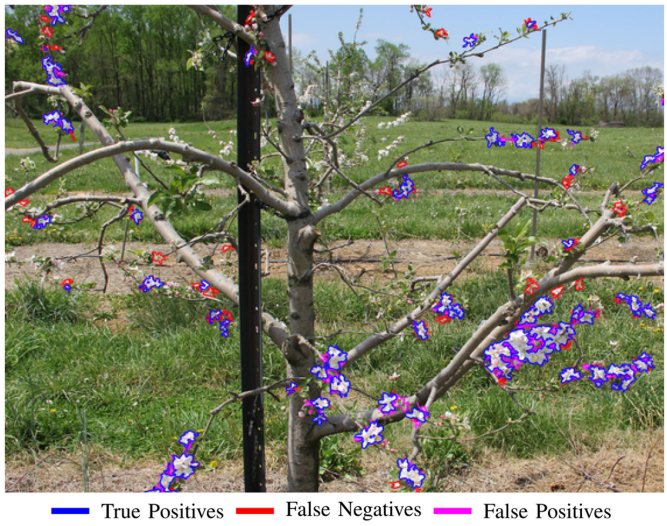

Fig. 6. **Best viewed in color.** Examples of flower detection in one image composing the *AppleA* dataset.

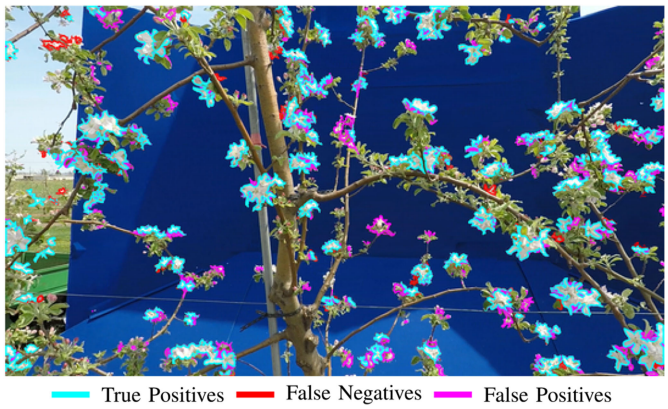

Fig. 7. **Best viewed in color.** Examples of flower detection in one image composing the *AppleB* dataset.

TABLE II
HSV STATISTICS OF IMAGES COMPOSING EACH DATASET

	H $[0-360°]$		S [%]		V [%]	
Dataset	μ_H	IQR_H	μ_S	IQR_S	μ_V	IQR_V
AppleA	74.6	49.3	32.9	24.3	53.7	30.2
AppleB	219.6	21.1	88.6	44.3	47.1	16.9
Peach	223.8	199.9	11.8	20.7	42.3	46.6
Pear	85.9	178.8	16.4	23.4	42.4	20.8

TABLE III
HSV STATISTICS OF FLOWERS COMPOSING EACH DATASET

	H $[0-360°]$		S [%]		V [%]	
Dataset	μ_H	IQR_H	μ_S	IQR_S	μ_V	IQR_V
AppleA	136.6	205.5	6.3	9.8	77.3	24.3
AppleB	56.3	80.2	7.5	9.8	86.7	23.1
Peach	325.2	26.7	21.2	13.3	50.2	13.7
Pear	215.4	173.2	5.9	5.9	84.7	22.4

summarize the differences among datasets in terms of the statistics of the HSV color components, where μ stands for mean values and IQR for interquartile ranges.

Regarding the flower characteristics, apple blossoms are typically white, with hue components spread in the whole spectrum (high IQR_H) and low saturation mean. Flowers composing the *AppleB* dataset present higher brightness (μ_V), while peach flowers show a pink hue centered on $\mu_H = 325°$, with higher saturation and lower value means. Moreover, pear flowers are slightly different in terms of color (greener) and morphology, as illustrated in Fig. 9.

A. Labeling

Image annotation for segmentation tasks is a laborious and time-consuming activity. Labels must be accurate at pixel-level, otherwise both supervised training and the evaluation of segmentation techniques are compromised. Most existing annotation tools rely on approximating segmentations as polygons, which provide ground truth images that frequently lack accurate adherence to real object boundaries [32].

We opted for a labeling procedure that combines freehand annotations and RGR refinement [32]. Using a tablet, the user draws traces on regions of the image that contain flowers, indicating as well hard negative examples when necessary. These traces indicate high-confidence segmentation points, which are used as reference by RGR to segment the remaining parts of the image. Fig. 5 shows an example of a ground truth segmentation obtained using this procedure.[1]

V. EXPERIMENTS AND RESULTS

We aim at a method capable of accurate multi-species flower detection, regardless of image acquisition conditions and without the need for dataset-specific training or pre-processing. To verify that our method satisfies all these requirements, we performed experiments on the four different datasets described in Section IV while only using the *AppleA* dataset for training.

We adopt as the main baseline our previous model described in [11], which highly outperformed existing methods by employing the Clarifai CNN architecture to classify individual superpixels. We therefore refer to that model as SPPX+CLARIFAI and to our new method as DEEPLAB+RGR. We also compare our results against a HSV-BASED method [15] that segments images based only on HSV color information and size filtering according to threshold values optimized using grid-search.

[1]We will make the annotation tool publicly available as future work.

TABLE IV
SUMMARY OF RESULTS OBTAINED FOR EACH METHOD

		IoU	$\mathbf{F_1}$	Recall	Precision
AppleA	HSV-BASED	28.0%	43.7%	56.5%	35.7%
	SPPX+CLARIFAI	51.3%	67.8%	73.2%	63.1%
	DEEPLAB+RGR	**71.4%**	**83.3%**	**87.7%**	**79.4%**
AppleB	HSV-BASED	49.3%	66.0%	58.9%	75.1%
	SPPX+CLARIFAI	50.6%	67.2%	68.4%	66.1%
	DEEPLAB+RGR	**63.0%**	**77.3%**	**91.2%**	**67.1%**
Peach	HSV-BASED	0.1%	1.4%	1.4%	1.6%
	SPPX+CLARIFAI	49.1%	67.2%	71.3%	61.2%
	DEEPLAB+RGR	**59.0%**	**74.2%**	**64.8%**	**86.8%**
Pear	HSV-BASED	39.7%	56.8%	65.6%	50.1%
	SPPX+CLARIFAI	40.5%	57.6%	49.6%	68.7%
	DEEPLAB+RGR	**75.4%**	**86.0%**	**79.2%**	**94.1%**

All three methods were tuned using the *AppleA* training dataset, with differences in the pipeline for transfer learning. For the three unseen datasets, the SPPX+CLARIFAI relies on a pre-processing step that enhances contrast and removes the different backgrounds present in the images. Our new method DEEPLAB+RGR does not require any pre-processing. Instead, it employs the same pipeline regardless of the dataset, requiring only adjustments in portrait size. As summarized in Table I, images composing the *AppleA* dataset have resolution $4.3\times$ larger than images in the other three datasets. Thus, we split images in these datasets into portraits of 155×155 pixels, rather than the 321×321 pixels portraits used for *AppleA*.

The quantitative analysis of segmentation accuracy relies on precision, recall, F_1 and intersection-over-union (IoU) metrics [29] computed at pixel-level, instead of the superpixel-wise metrics used in our previous work. Table IV summarizes the results obtained by each method on the different datasets.

Our new model outperforms the baseline methods for all datasets evaluated, especially in terms of generalization to unseen datasets. By combining a deeper CNN architecture and the RGR refinement module, DEEPLAB+RGR improves both prediction and recall rates in the validation *AppleA* set by more than 15%. Fig. 6 provides a qualitative example of flower detection accuracy in this dataset.

As Fig. 7 illustrates, images composing the *AppleB* dataset present a higher number of flower buds and illumination changes, especially in terms of sunlight reflection by leaves. Despite the larger variance in comparison to the previous dataset, the performance obtained by DEEPLAB+RGR surpasses 77% in terms of F_1.

Results obtained for the *Peach* dataset demonstrate the limitation of color-based methods and two important generalization characteristics of our model. The HSV-BASED method is incapable of detecting peach flowers, since their pink color is very different from the white apple blossoms used for training. On the other hand, our method presents F_1 near 75%, indicating that it can properly detect even flowers that differ to a great extent from apple flowers in terms of color. Moreover, images composing this dataset are characterized by a cloudy sky and hence poorer illumination. Most cases of false negatives correspond to flower buds, due to the lack of such examples in the training dataset. As illustrated in Fig. 8, poor superpixel segmentation leads the SPPX+CLARIFAI approach to incorrectly classify parts of the sky

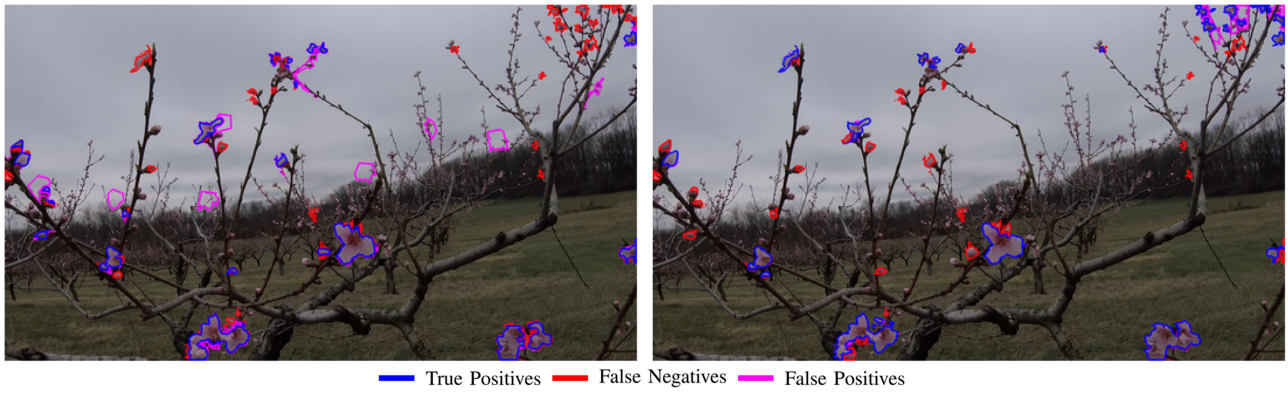

Fig. 8. **Best viewed in color.** Examples of flower detection in one image composing the *Peach* dataset. *Left:* detections provided by the SPPX+CLARIFAI method. *Right:* detections obtained with our new DEEPLAB+RGR method.

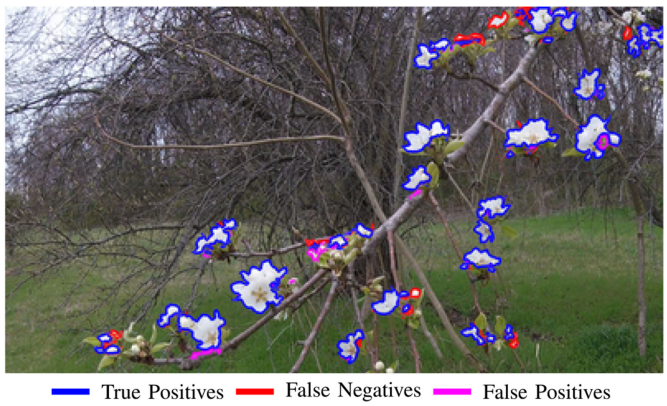

Fig. 9. **Best viewed in color.** Examples of flower detection in one image composing the *Pear* dataset.

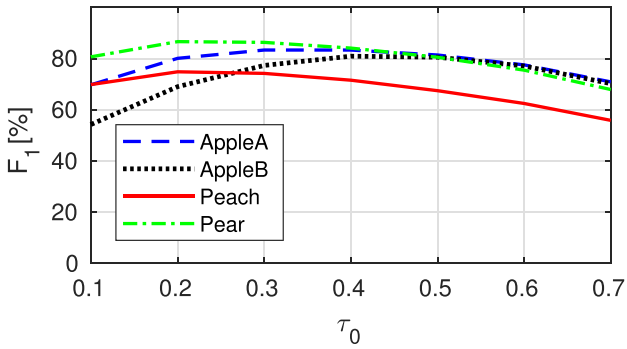

Fig. 10. Segmentation performance in terms of F_1 measure on each dataset according to the parameter τ_0.

as flowers. This problem is overcome by our new model, which greatly increases precision rates to above 80%.

Furthermore, the high recall rate provided by DEEPLAB+RGR in the *Pear* dataset demonstrates its robustness to slight variations in both flower morphology and color. As shown in Fig. 9, similar to the *Peach* dataset, these images also present a cloudy background. In addition to that, their background is characterized by a high level of clutter caused by the presence of a large number of branches. These high texture components compromise the background removal model used by SPPX+CLARIFAI. Still, the DEEPLAB+RGR method provides a very accurate detection of flowers, with precision above 90%.

The results obtained by our method for *AppleB*, *Peach* and *Pear* datasets can be further improved by adjusting the parameter τ_0 used for final classification and refinement. As summarized in Fig. 10, increasing τ_0 from 0.3 to 0.5 increases in 3% the F_1 performance on *AppleB*, reaching both recall and precision levels around 80%. For the *Peach* dataset, decreasing τ_0 to 0.2 increases the recall rate to above 70%. Such adjustment can be carried out quickly through a simple interactive procedure, where τ_0 is chosen according to its visual impact on the segmentation of a single image.

In terms of inference time, the current implementation of our algorithm on an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz (62GB) with a Quadro P6000 GPU requires on average 50 seconds to evaluate each high-resolution image composing our datasets. Around 5 seconds are required to save portraits as individual files and load their corresponding prediction scores, a process that can be simplified by generating portraits directly within the neural network framework.

VI. CONCLUSION

We have presented a novel automated approach for flower detection, which exploits state-of-the-art deep learning techniques for semantic image segmentation. The applicability of our method was demonstrated by its high flower segmentation accuracy across datasets that vary in terms of illumination conditions, background composition, image resolution, flower density and flower species. Without any supervised fine-tuning or image pre-processing, our model trained using only images of apple flowers succeeded in generalizing for peach and pear flowers, which are noticeably different in terms of color and morphology.

In the future, we intend to further improve the generalization capabilities of our model by training and evaluating it on multi-species flower datasets. We ultimately aim at a completely autonomous system capable of online bloom intensity estimation. The current implementation of our model can evaluate high-resolution images of complete trees an order of magnitude

faster than human workers. While in this work we are not creating maps of flowers at the block level, this method will scale well for precision agricultural applications such as predicting thinning spray treatments and timing.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA Corporation for the donation of the GPU used for this research. Mention of trade names or commercial products in this publication is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the U.S. Department of Agriculture. USDA is an equal opportunity provider and employer.

REFERENCES

[1] C. Forshey, "Chemical fruit thinning of apples," New York State Agricultural Experiment Station, Geneva, Cornell Univ., Ithaca, NY, USA, 1986.

[2] H. Link, "Significance of flower and fruit thinning on fruit quality," *Plant Growth Regulation*, vol. 31, no. 1/2, pp. 17–26, 2000.

[3] A. Gongal, A. Silwal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Apple crop-load estimation with over-the-row machine vision system," *Comput. Electron. Agriculture*, vol. 120, pp. 26–35, 2016.

[4] N. Zhang, M. Wang, and N. Wang, "Precision agriculture—A worldwide overview," *Comput. Electron. Agriculture*, vol. 36, no. 2/3, pp. 113–132, 2002.

[5] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Ben-Shahar, "Computer vision for fruit harvesting robots–state of the art and challenges ahead," *Int. J. Comput. Vis. Robot.*, vol. 3, no. 1/2, pp. 4–34, 2012.

[6] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Sensors and systems for fruit detection and localization: A review," *Comput. Electron. Agriculture*, vol. 116, pp. 8–19, 2015.

[7] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[8] S. W. Chen *et al.*, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 781–788, Apr. 2017.

[9] M. Dyrmann, A. K. Mortensen, H. S. Midtiby, and R. N. Jørgensen, "Pixel-wise classification of weeds and crops in images by using a fully convolutional neural network," in *Proc. Int. Conf. Agricultural Eng.*, Aarhus, Denmark, 2016, pp. 26–29.

[10] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, "Deep learning for plant identification using vein morphological patterns," *Comput. Electron. Agriculture*, vol. 127, pp. 418–424, 2016.

[11] P. A. Dias, A. Tabb, and H. Medeiros, "Apple flower detection using deep convolutional networks," *Comput. Ind.*, vol. 99, pp. 17–28, Aug. 2018.

[12] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[13] P. A. Dias, A. Tabb, and H. Medeiros, "Data from: Multi-species fruit flower detection using a refined semantic segmentation network," 2018. [Online]. Available: https://data.nal.usda.gov/dataset/data-multi-species-fruit-flower-detec tion-using-refined-semantic-segmentation-network

[14] A. D. Aggelopoulou, D. Bochtis, S. Fountas, K. C. Swain, T. A. Gemtos, and G. D. Nanos, "Yield prediction in apple orchards based on image processing," *Precision Agriculture*, vol. 12, no. 3, pp. 448–456, 2011.

[15] M. Hočevar, B. Širok, T. Godeša, and M. Stopar, "Flowering estimation in apple orchards by image analysis," *Precision Agriculture*, vol. 15, no. 4, pp. 466–478, 2014.

[16] K. R. Thorp and D. A. Dierig, "Color image segmentation approach to monitor flowering in lesquerella," *Ind. Crops Products*, vol. 34, no. 1, pp. 1150–1159, 2011.

[17] R. Horton, E. Cano, D. Bulanon, and E. Fallahi, "Peach flower monitoring using aerial multispectral imaging," *J. Imag.*, vol. 3, no. 1, 2016.

[18] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[19] C. Hung, J. Nieto, Z. Taylor, J. Underwood, and S. Sukkarieh, "Orchard fruit segmentation using multi-spectral feature learning," *IEEE Proc. Int. Conf. Intell. Robots Syst.*, 2013, pp. 5314–5320.

[20] S. Bargoti and J. P. Underwood, "Image segmentation for fruit detection and yield estimation in apple orchards," *J. Field Robot.*, vol. 34, no. 6, pp. 1039–1060, 2017.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[22] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Appl. Soft Comput.*, vol. 70, pp. 41–65, 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. 3rd IAPR Asian Conf. Pattern Recognit.*, 2014, pp. 730–734.

[25] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1520–1528.

[26] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for scene segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[27] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 447–456.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[29] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2015.

[30] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[31] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 109–117.

[32] P. A. Dias and H. Medeiros, "Semantic segmentation refinement by Monte Carlo region growing of high confidence detections," 2018. [Online]. Available: https://arxiv.org/abs/1802.07789

[33] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and stuff classes in context," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1209–1218.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[35] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia.*, 2014, pp. 675–678.