IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 3, 2020, accepted June 11, 2020, date of publication June 18, 2020, date of current version July 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3003415

3DBunch: A Novel iOS-Smartphone Application to Evaluate the Number of Grape Berries per Bunch Using Image Analysis Techniques

SCARLETT LIU[1,2], (Member, IEEE), XIANGDONG ZENG[1], AND MARK WHITTY[2], (Member, IEEE)

[1]School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China
[2]School of Mechanical and Manufacturing Engineering, UNSW Sydney, Sydney, NSW 2052, Australia

Corresponding author: Scarlett Liu (scarlett.liu@csu.edu.cn)

This work was supported in part by the UNSW and in part by the Central South University.

ABSTRACT Evaluating the number of berries per bunch is a vital step of grape yield estimation in viticulture but is a labour intensive task for traditional manual measurement. Therefore, this paper develops a novel smartphone application for counting berries automatically from a single image. The application, called 3DBunch, acquires images from the camera or the album on a smartphone, and then estimates the number of berries by a reconstructed 3D bunch model based on the proposed image analysis techniques that are embedded in the developed iOS app. It also presents features of visualising the statistics of the reconstructed bunch, which including of the distribution of detected berry size in pixels and the total number of berries. It also has the capability of presenting sampling related information, which includes the person who conducted the sampling, the location of samples, the dates of sampling, the variety, farm, vineyards *etc*. The application was evaluated both on a simulator on a commercial computer and an iPad mini 4. By analysing 291 bunch images from two varieties the app achieved an accuracy of 91% regarding berry counting per bunch. Additionally, the computational time consumed to process 100 images on iPad mini 4 was studied and returned an average of 7.51 seconds per image. Obtaining these results with only a smartphone and a small backing board for capturing a photo with a single bunch, 3DBunch provides an efficient way for farmers to count berries in-vivo and it is available in the iOS App Store.

INDEX TERMS Berry counting, bunch analysis, image analysis, iOS application, yield estimation.

I. INTRODUCTION

In viticulture, agronomic parameter analyses based on a large number of samples are essential to guarantee the accuracy of grape yield estimation. This is especially true in Australia, where the labour force is extremely costly and difficult to arrange during the grape growing season. According to one summary report released by Wine Australia [1], the total vineyard area in 2019 was 146, 128 ha, ranking it the sixth largest wine producer in the world. There are 6,251 grape growers and 163,790 employees (full and part-time). The wine business contributes over $45.5 billion annually to the Australian economy. The contract set between the grape growers and winemakers has to be done before the harvest each year, which means the harvest yield prediction is vital for the production chain. And one of the critical parameters that vineyard managers care about is the number of berries per bunch.

The associate editor coordinating the review of this manuscript and approving it for publication was Haiyong Zheng.

However, traditional manual approaches for grape yield estimation are not efficient enough, especially for counting the number of berries per bunch. This leads to a lack of samples and results in inaccurate grape yield estimation. Usually flowering occurs around November for the Southern Hemisphere regions and the harvest period starts from January and ends in March [2], depending on the areas and climate in the current year. Hence monitoring the growing status of bunches makes it one important vineyard management practice. Driven by this, the new technologies for grape production forecasting are developing rapidly with the assistance of image sensors, and mobile devices.

Image-based techniques for yield management gradually developed in recent years, and many researchers turn to image processing to find solutions. Seng *et al.* [3] established the GrapeCS-ML database for researchers to apply computer vision and machine learning in smart vineyards. Reis *et al.* [4] proposed a system to locate bunches of grapes and automatically distinguish between white and red grapes, which works at night with artificial illumination to avoid

brightness variation. Font *et al.* [5] presented an automatic method based on detection of the specular reflection from the spherical surface of the grapes. Pixel-based segmentation and converting area and estimated volume of clusters of grapes to yield estimation according to specific calibration curves were also investigated in grape production forecasting [6]. An all-terrain vehicle was used by Aquino *et al.* [7] to capture vine images, mathematical morphology and pixel classification were used to analyse the image. However, aforementioned methods can only be used at night hence they are not as convenient and effective as the methods that can be utilized during daytime. Nuske *et al.* [8] estimated yield across the entire vineyard by applying the radial symmetry transform to detect potential berry locations and k-Nearest Neighbors algorithm to identify the potential locations which have similar appearance to grape berries. Zabawa *et al.* [9] presented a method based on fully convolutional neural networks to count single berries in images. But the size of berries was not investigated in these works. Mirbod *et al.* [10] presented a vehicle mounted imaging system to measure the diameter of berries and generate a map of the berry size variability. However this method is not convenient to be directly applied by viticulturists because of the specialized capturing platform.

Nowadays, affordable smartphones can provide excellent computing performance and photographic capabilities. A wide range of options which were previously infeasible to collaborate with the phone can be developed into mobile devices for specialized applications used in agriculture. Hernández-Hernández *et al.* [11] developed a smartphone application called pCAPS to obtain information on plants and soil, including percentage of green cover, number of objects and so on. Laamrani *et al.* [12] presented a mobile device application to access crop residue cover in the field after harvest, and compared it against two point counting approaches to prove the developed application was valid.

Smartphone applications are also utilized in citrus and apple cultivation. Gong *et al.* [13] used an Android mobile phone to estimate the yield of citrus on an individual tree by image processing, and Cubero *et al.* [14] developed an application to facilitate the measurement of citrus colour index. In order to estimate apple yield in a fast, convenient and low cost way, Qian *et al.* [15] proposed a distribution framework based on smartphone and server client for image acquisition, yield prediction, data processing and model calculation and generate a yield map to identify the space distribution of the yield. Smartphone applications can also be applied to assess rice nutritional status [16] and detect damage caused by blue mould on tobacco leaves [17]. Apart from these, the information provided by smartphone applications can help farmers make better decision, such as scientific e-advisories for menthol mint crop [18], time of applying fungicide in late-season soybean diseases management [19], irrigation schedule [20], [21] and so on. Integrated with a sensor stack, smartphone apps can be implemented in wider agriculture areas such as monitoring crop information [22], controlling greenhouse environment [23] and photographing suspicious activity to prevent farmers' losses [24].

There are not many mature mobile applications used in viticulture yet. Ang *et al.* [25] investigated the feasibility of applying smartphone in berry quality assessment as an image processing tool. Grossetete *et al.* [26] presented a smartphone application to count berries based on the specular reflection of the berry surface; yet it does not appear in any app store right now and appears far out of date. Aquino *et al.* [27] developed an new algorithm to count berry number per bunch in digital images and extended their work into an app [28]. This work was released followed their previous flower detection app [29]. However, only green berries were mentioned in their work and the scalability of their applications are not large. Most importantly, their apps are no longer in related App Store for usage or direct comparison. Schmidtke [30] and Rahaman *et al.* [31] presented an application to sample visible berries and analyse the volume and hue distributions of detected berries according to single grape bunch images photographed on the vine, but total number of the berries per bunch was not mentioned, which is one of the key parameters for grape yield estimation.

Therefore, in this paper, a novel app, called 3DBunch, is presented to contribute a fast in-field application to assist grape yield estimation. The performance and computational efficiency of this app is tested and reported regarding estimating the number of both green and purple grape berries per bunch from single images. The number of berries in a bunch image is strongly correlated to the actual number in the real bunch [32], and the radii distribution of grape berries and other information related to each sampled bunch can be logged in a cloud server by 3DBunch.

This paper mainly focuses on the usability of the designed app, testing and validating the performance of it. It also add other features, which including data logging, cloud server and data visualization based on processed images. This work explicitly provided an easy-to-use mobile phone application for allowing grape farmers, vineyard technicians and viticulturists to count berries, record data, view data, and retrieve data from the cloud server.

The rest of this paper is organised as follows: Section II and Section III present the theoretical details of 3DBunch app, which includes the design and implementation of the app, and the internal algorithm we propose regarding 3D model reconstruction based on a single image. Section IV describes the datasets and devices that are utilized for validating 3DBunch. Performance of accuracy and computational efficiency analysis of 3DBunch are presented in Section V. Section VI lastly draws conclusions and makes recommendations for future work.

II. THE DESIGN AND IMPLEMENTATION OF 3DBunch

The 3DBunch app combines image acquisition, analysis, result display and storage to provide viticulturists with a comprehensive tool for estimating the number of berries per bunch, and it was developed for smartphones powered

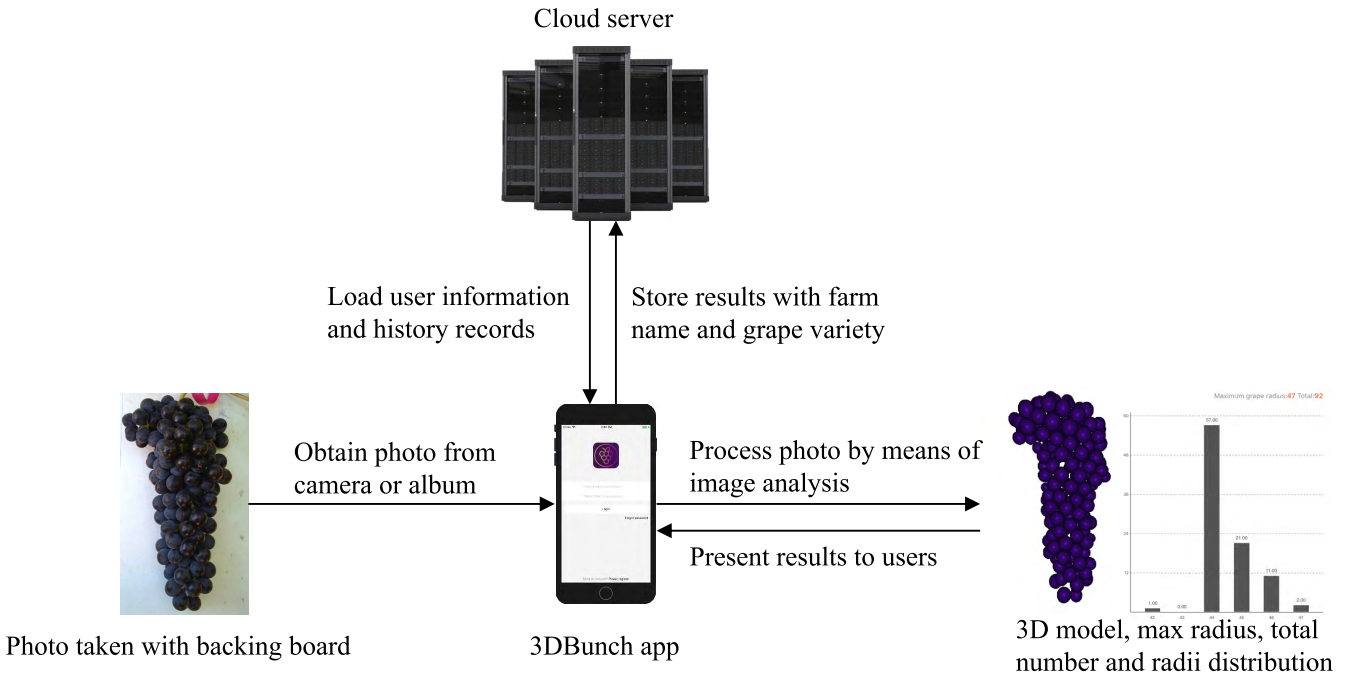


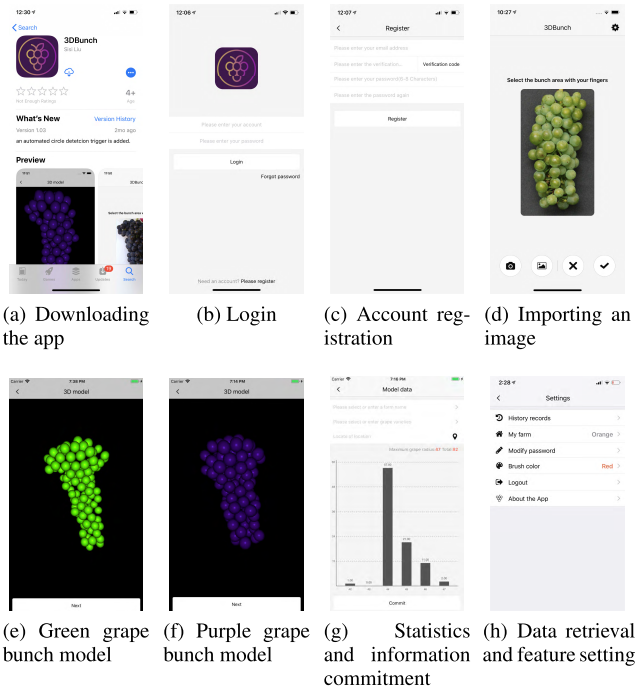
FIGURE 1. Overview of 3DBunch.

with iOS [33] version 11.4 and above. In order to facilitate new users to familiarize themselves with the application quickly and master the operation in a short time when they first use 3DBunch, the application selected model-view-controller(MVC) [34] as the software architecture pattern. It allows end-users to re-do any step while operating the app. It updates the parameters and outcomes instantly and can be easily understood by any end-users.

As a tool for assisting grape berry counting, 3DBunch enable users to easily to capture or upload existing photos locally. It will automatically returned the processed model and the estimated berry number by the proposed internal algorithm after the image is pre-processed properly. It also allows end-users to log, store, and retrieve their data from the cloud server. This includes the details of account info, vineyard, block, variety, dates, sampling site, GPS coordinates and processed information extracted from the image, which contains the berry number and berry size distribution (in pixels). The functionality of 3DBunch has been clearly demonstrated in Fig. 2. The internal algorithms regarding image processing, 3D model reconstruction and visualization are embedded locally instead of employing a cloud server. The reason for this is that end-users can obtain the results instantly even in the absence of sufficient internet connectivity. It also collaborates with the sampling procedure, which is the main method of manual yield estimation in vineyards, by recording aforementioned data for each photo.

A. INTERFACE DESIGN

New users can search "3DBunch" in Apple App Store and download it onto iOS based smart mobile devices, as demonstrated in Fig. 2a. Once the app is installed,



(a) Downloading the app (b) Login (c) Account registration (d) Importing an image

(e) Green grape bunch model (f) Purple grape bunch model (g) Statistics and information commitment (h) Data retrieval and feature setting

FIGURE 2. Interface design.

the initial page users are presented with is the registration page (Fig. 2b). New users can register a new account by clicking the bottom line in Fig. 2b and then a registration page (Fig. 2c) will show up and guide new users to finish the registration with a verification code from their nominated email account. To have an account is a starting point for utilizing this app since all processed data will be stored and associated with this account.

After the account registration, the user can either choose to take a live photo by the local camera or import an existing photo from a local album by clicking the first or second button at the bottom, see Fig. 2d. Allowing users to import a photo from the local album enables users to process collected photos captured by the default camera at a later time. After importing a proper photo (a grape bunch with a backing board filled with high-contrast color), four major processes will be applied to reconstruct a 3D grape bunch model, consisting of image cropping, color segmentation, edge detection, visible berry detection, and 3D bunch model reconstruction. The details of those operations are presented in Section II-B. Those four procedures can be conducted within approximately five seconds. Fig. 2e and 2f demonstrate two grape bunches with different colors which are rendered by the iOS rendering engine.

Fig. 2g illustrates the visualization of the reconstructed bunch and some statistics about the bunch. It includes the distribution of the berry size, the maximum berry size in pixels and the estimated berry number. The page also allows users to input some data associated with farm names and varieties. GPS location is automatically retrieved from the smart device. Otherwise users can input it manually or ignore it. After filling related information, users can click the commit button at the bottom of the page and the user will be taken back to the image acquisition page (Fig. 2d) after the commit is successful.

In addition, some feature settings of the app interface can be accessed by clicking the gear icon at the upper right corner of the image acquisition page, from where users can visit the historical records, modify passwords, and so on. The app also provides some proper instructions to guide users to take actions at some key steps (eg., Fig. 4b and 4f). In short, the developed app offers a user-friendly interface for all users based on the aforementioned features.

B. IMPLEMENTATION

A flowchart describing the implementation of 3DBunch is depicted in Fig. 3. Nine major steps are summarized as follows:

1) **Launch**. A splash screen will be displayed when 3DBunch is launched and it will show a 3DBunch logo, see Fig. 4.
2) **Login**. Users need to input their accounts and passwords when they access the app. User registration and password reset also can be conducted in the login interface, see Figures 2b and 2c.
3) **Image Acquisition**. Users can obtain images from the local camera or album (Fig. 2d). The acquired image will be showed at the middle of the interface, it also can be discarded and reacquired if the previous one was not satisfied.
4) **Grape Area Cropping**. This is step is not compulsory. Users can skip it if the captured grape bunch is the main object in view with a clean backing board. Otherwise users need to select the grape area with their fingers to avoid interference from other noise, such as branches, leaves and *etc.*, whose color is similar to the berries. The selection must be completed with one stroke, and the head and tail will be automatically connected in a loop. The background outside the area will turn to white for purple grapes and black for green grapes, see Fig. 4b.
5) **Color Selection**. Purple or green (Fig. 4c will be selected by users after grape area selection. They represent the color of grapes, and different selections will lead to different image preprocessing (purple for RGB split and green for L*a*b* split).
6) **Channel Selection**. Users can choose a channel from red, green and blue (Fig. 4e) for purple grapes or L*, a* and b* layer (Fig. 4d) for green grapes, in which the grape area has the most prominent attributes. Empirical results show that for green and purple grapes the a* channel and blue channel respectively are the most useful for segmenting the berries.
7) **Parameter Adjustment**. This step can be skipped since the threshold and radius are determined automatically by 3DBunch. However, for achieving a better performance for some special cases, users can adjust the distance between the two sliders to determine more appropriate threshold and radius. Parameter adjustment (Fig. 4g) allows users to check the circle detection visually and fine-tune the results. The optimized threshold and radius are determined based on the following three factors visually:
   - The outline formed by the detected circles should fit the edges of grape area as well as possible.

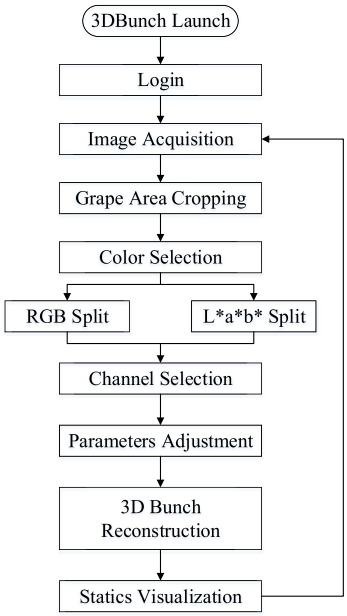
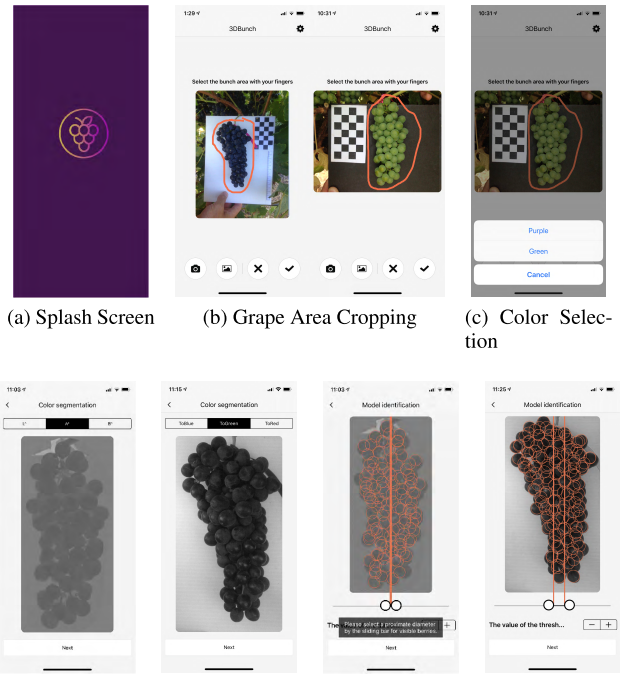


FIGURE 3. Flowchart of the 3DBunch app.



(a) Splash Screen (b) Grape Area Cropping (c) Color Selection

(d) Channel Selection - Green (e) Channel Selection - Purple (f) Radius Calculation by default (g) Parameter Adjustment

FIGURE 4. **Main screenshots of 3DBunch.**

- The size of detected circles should coincide with that of berries on the image as well as possible.
- The radii of detected circles should have different values (users will see circles with different sizes).

8) **3D Bunch Reconstruction**. The internal algorithm will be operated automatically by reconstructing the 3D grape bunch model. The reconstructed bunch will be rendered to green (Fig. 2e) or purple (Fig. 2f) by GLKit iOS [35]. GLKit was found to provide faster rendering than openGL. Users can rotate, zoom in or zoom out the model with their fingers, see Figures 2e and 2f.
9) **Statistics Visualization and Information Commit**. The maximum radius, the number of detected berries and the histogram of radii distribution will be presented to the users. The results can be saved after inputting the farm name and grape variety. The GPS location of the taken photo will be automatically retrieved from the mobile device, see Fig. 2g. The interface will return to image acquisition after the commit is completed (Fig. 2h).

In short, 3DBunch is fully equipped by the iOS framework. The main advantage of iOS application is the uniformed specification of mobile devices. That leads to easier version control compared with Android Application System, which requires more version testing according to various mobile device framework. The auto exposure and white balance of its camera setting saves effort for end users to handle certain condition of taking an adequate photo. The 3DBunch is available in the iOS App Store now.

III. INTERNAL IMAGE PROCESSING ALGORITHM OF 3DBunch

3DBunch is an original application developed based on iOS devices to estimate the number of berries per bunch by analysing an image containing a single bunch. Its internal algorithm for image processing and 3D grape bunch reconstruction is illustrated in the flowchart in Fig. 5.

Inspired by the work mentioned in paper [32], the proposed algorithm first localizes the bunch as the initial Region of Interest (ROI) after the manual crop operation. The reason we set this manual cropping is that it is difficult to get a clean image which only contains one bunch with a backing board; specifically the problem of identifying the peduncle is challenging when there are multiple sub-bunches, tendrils or wings. The dilemma is that user cannot hold the backing board behind the grape perfectly without the interruption of branches, leaves and *etc.* (see Fig. 6a), while holding a mobile device to take a photo. So the backing board is either too big to fit or too small for avoiding interference with one hand. For some 'user-friendly' scenarios (see Fig. 6b), the manual crop step can be skipped straight away because of very exposed bunches. The real testing situation for both scenarios are illustrated in Fig. 6.

Once the image is correctly acquired, 3DBunch applies the proposed algorithm that is improved from the method presented by Liu *et al.* [32] to estimate berries by adapting it to the iOS system. The internal algorithm can be divided into four major steps from the perspective of image analysis and the parts amended from the original work are explained as follows:

1) **Grape area extraction**. The region of interest (ROI) extraction is conducted by Otsu's method [36] in the original algorithm. Catering for better user-experience, binarization with fixed threshold values is set in the process of ROI extraction (Fig. 8b) after selecting manually an appropriate channel (mentioned in Section II-B). A value of 95 is used for the purple bunches and 120 is set for the green ones. The outline of the bunch (Fig. 8c) is then extracted by finding its contour in the captured or imported image.
2) **Finding berries at edge**. Determining the threshold and the range of radius of berries to apply Hough transform is a vital step in berry counting. Even though the radius of circles are determined automatically in original paper [32] the method proposed in that paper is too complicated to implement with iOS system. So we simplified this part for 3DBunch app.
Once the outline has been extracted, the range of radius will be initialized to apply Hough Transform [37] for detecting circles which fit the outline well (Fig. 8d). The minimum radius and max radius are calculated as:

$$R_{min} = w/2n_{max} \quad (1)$$

$$R_{max} = R_{min} + S \quad (2)$$

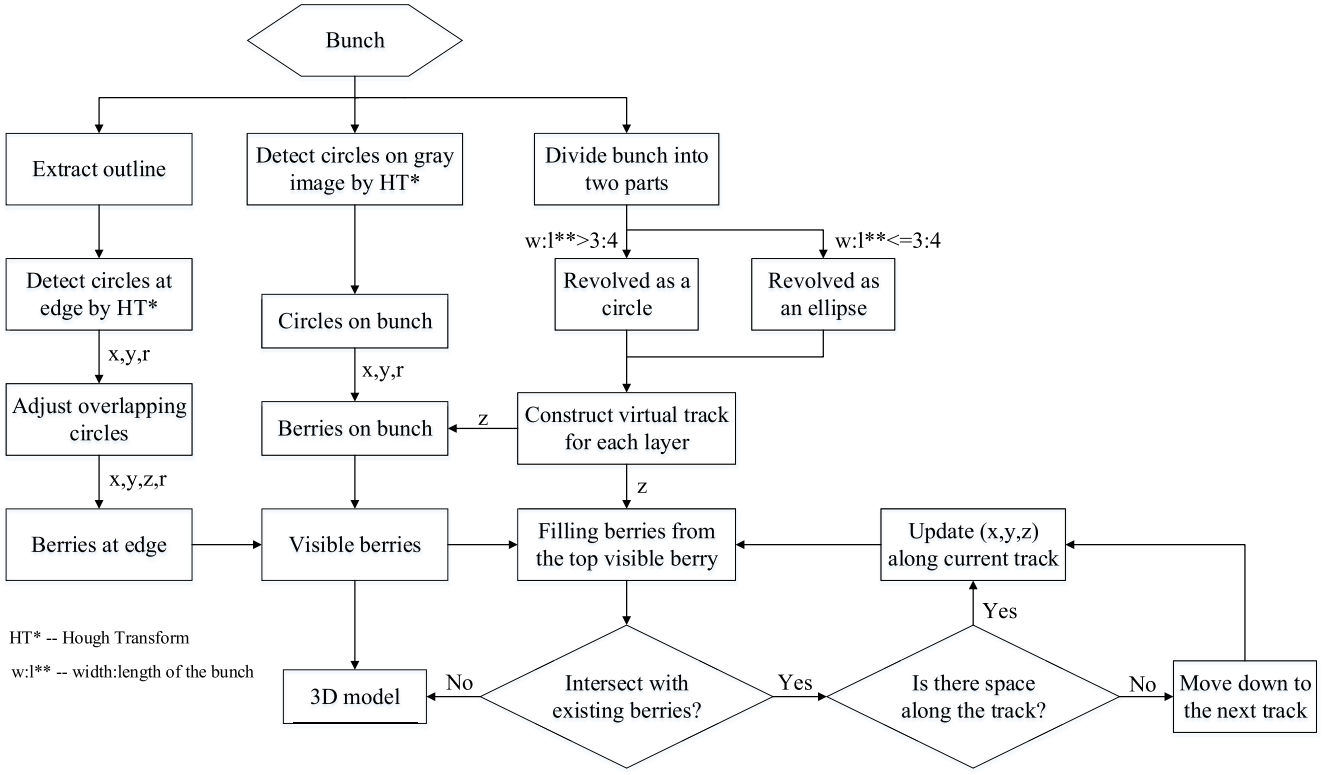
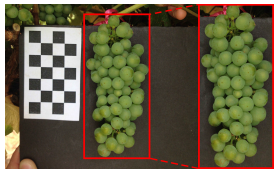


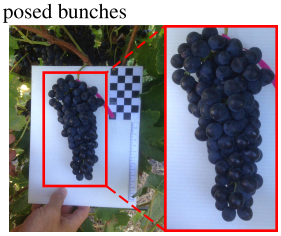
FIGURE 5. Flowchart of the internal algorithm of 3DBunch app.

(a) Scenario 1, intertwined and overlapping bunches

(b) Scenario 2, isolated and exposed bunches



(c) Crowded bunch, short stem, making it difficult to put a backing board behind the grape bunch, even more difficult to get the whole bunch covered by the backing board.



(d) Isolated bunch, long stem, easy to put a backing board behind the grape bunch

FIGURE 6. Main screenshots of 3DBunch.

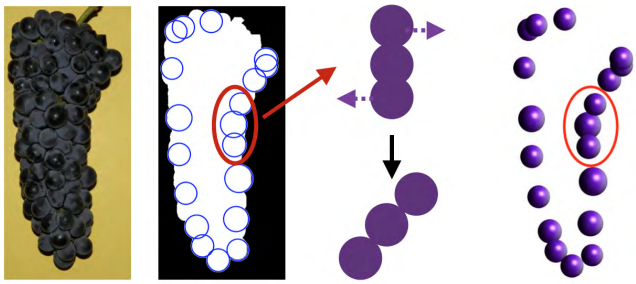
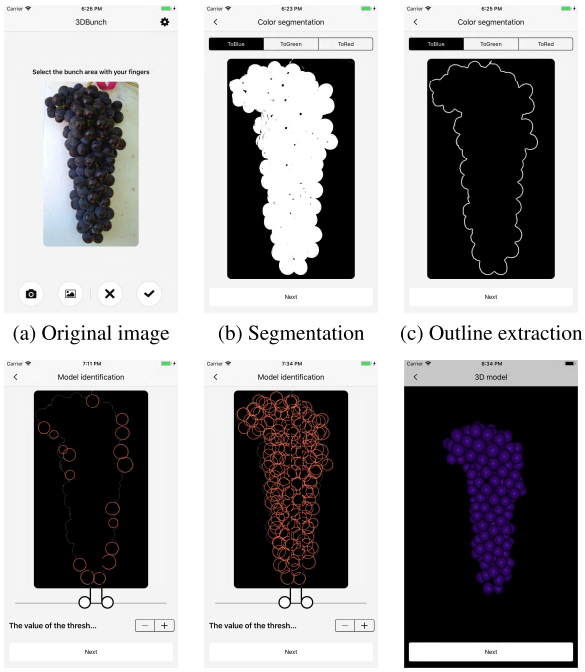


FIGURE 7. Adjusting overlapping berries at edge.

where w is the width of the bounding box of the outline, and n_{max} refers to the max number of berries in a line. According to the datasets tested, a value of $n_{max} = 15$ is an appropriate experimental value. If R_{min} is too small to apply the Hough transform, it will be incremented until the Hough transform returns a result. S refers to the range of the radii, which is defined as 10 according to empirical statistical analysis. This is how initial berry radius is set by default (see Fig. 4f). Users also can adjust the radius for better results, which mentioned in Step 7, Section II-B. The locations (x and y coordinates in the image plane) and radii of those detected circles are utilized to form the 3D grape bunch model by placing corresponding spheres in a plane normal to the direction of view. As for the overlapping spheres, they are moved forward and backward along the z axis (normal to the paper plane) respectively until there is no overlap. This locates the berries at the edge.

3) **Visible berry detection**. A Hough transform is applied again on the gray image of the bunch to detect all internal visible berries (Fig. 8e) by providing corresponding x, y coordinates and radii of detected circles at the edge and the berries at the edge. The bunch is divided into two parts if its width-length ratio is larger than 3:4. A watershed is formed at 3/4 of its length down the



(a) Original image (b) Segmentation (c) Outline extraction

(d) Berry fitting at the edge (e) Visible berry fitting (f) Reconstructed 3D model

FIGURE 8. Steps of 3DBunch processing bunch image: (b) - (e) are not shown in the release version of 3DBunch in Apple store but are shown in the simulator (mentioned in Section IV) for debugging and demonstration purposes.

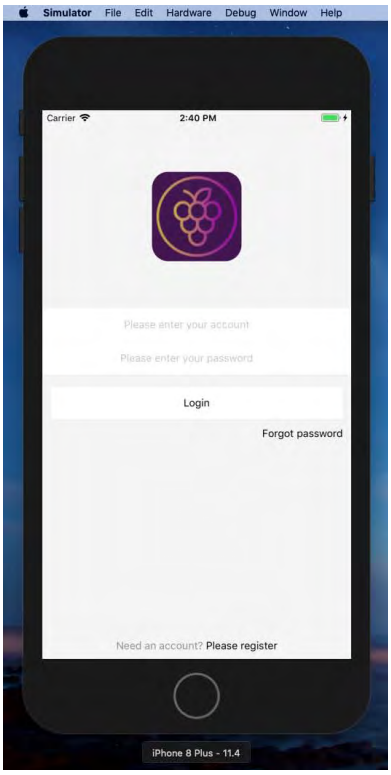


FIGURE 9. An iPhone 8 Plus simulator runs on a Macbook Air.

longitudinal axis. This value is defined based on the available data. Then the upper part is revolved about the longitudinal axis through its middle as an ellipse and z value for each berry is calculated as:

$$z = \sqrt{|(1 - \frac{(x - x_t)^2}{(l - r)^2})(l_e - r)|} \tag{3}$$

the lower part is revolved as a circle, and z value is calculated as:

$$z = \sqrt{|l_r^2 - (x - x_t)^2|} \tag{4}$$

where l is defined as:

$$l = \frac{x_l - x_f + 1}{2} \tag{5}$$

x_f and x_l refer to the x coordinate of fist and last pixel of the transversal which crosses the center of circle related to visible berry. Then x_t is formulated as:

$$x_t = x_f + l \tag{6}$$

And l_e is defined as:

$$l_e = \eta \cdot l \tag{7}$$

where η refers to the reduction ratio of the length when the section is revolved as an ellipse and its value is 5/6.

4) **Filling berries in a 3D model**. The radius of each populated berry is randomly chosen according to the normal distribution of radii from all detected visible berries and is defined as:

$$r_i = \alpha(m_u - m_l) + \mu \tag{8}$$

where α is a random number from 0 to 1, for radii of existing berries, m_u and m_l refer to upper bound and lower bound of confidence interval with 0.95 confidence level. The process begins from the top detected berry. Then a new berry is placed along the virtual track, which is generated by rotating the current width of the detected bunch about a vertical axis at a regular interval (1 degree). So the center (x and z coordinates) of each populated berry is calculated as:

$$x_i = x_t + (l - r_i) \cdot \cos(\frac{\theta}{180} \cdot \pi) \tag{9}$$

$$z_i = \begin{cases} (l_e - r_i) \cdot \sin(\frac{\theta}{180} \cdot \pi) & \text{ellipse section} \\ (l - r_i) \cdot \sin(\frac{\theta}{180} \cdot \pi) & \text{circle section} \end{cases} \tag{10}$$

value θ varies from 1 to 360. It is defined when there is no intersection with any existing sphere. This process moves down by a defined step size (two pixels) and repeats until it reaches the bottom of the bunch and y_i for each berry is defined during this process. Then, based on the position and radii of detected circles, a 3D grape bunch model is reconstructed (Fig. 8f).

IV. DATA SCOPE AND EXPERIMENTAL DESIGN

A. MOBILE DEVICES

Testing for validating 3DBunch's performance were conducted on both a simulator and real mobile devices. An iPhone 8 Plus (iOS version 11.4) was simulated on a MacBook Air (see Fig. 9) and for the real mobile device,

TABLE 1. Relevant features of the device for evaluating the performance of 3DBunch.

Device	Feature				
	Price/Dates	CPU	GPU	RAM	Operating system
iPad mini 4	$399 / 2018	Apple A8	PowerVR GX6450	2G LPDDR3	iOS12.3.1
iPhone 8 Plus	$799 / 2017	Apple A11	Series8XT	3G LPDDR4	iOS11
MacBook Air	$1120 / 2013	Intel Core i5	Intel HD Graphics 5000	4G DDR3	macOS Mojave

TABLE 2. Details of datasets for validating 3DBunch app.

Data-set	Number of images	in/ex-vivo	Variety	Colour	E-L stage[38]	Smartphone	Resolution[pixels]
D2	73	ex-vivo	Shiraz	purple	38 Harvest	LG G3	4160*2340
D4	44	in-vivo	Shiraz	purple	38 Harvest	iPhone 5	2448*3264
D5	56	in-vivo	Shiraz	purple	38 Harvest	iPhone 5	1536*2048
D6	63	in-vivo	Shiraz	green	31 Berries pea-sized	iPhone 4S	2448*3264
D7	56	in-vivo	Chardonnay	green	33 Berries still hard and green	iPhone 4S	2448*3264

TABLE 3. Performance of 3DBunch when comparing estimated and actual berry numbers.

Dataset	Device	Colour	E-L Stage	R^2	$e_a(\%)$	$e_p(\%)$
D2	Simulated iPhone 8 Plus	purple	38 Harvest	0.90	7.69	-5.36
D4	iPad mini 4	purple	38 Harvest	0.96	7.69	-0.79
D5	Simulated iPhone 8 Plus	purple	38 Harvest	0.98	5.03	0.81
D6	Simulated iPhone 8 Plus	green	31 Berries pea-sized	0.98	5.26	2.17
D7	iPad mini 4	green	33 Berries still hard and green	0.93	6.12	3.22

an iPad mini 4 was tested. Relevant specifications of the devices can be found in Table 1.

B. DATASETS

For the datasets, 291 bunch images from two varieties were acquired and analysed, details of each dataset are listed in Table 2. The cluster images from D4 and D7 were detected by 3DBunch installed on iPad mini 4, and those from D2, D5 and D6 were used to test the 3DBunch on the simulator.

C. EVALUATION INDICATORS

1) INDICATORS FOR THE PROPOSED ALGORITHM PERFORMANCE

For each photographed bunch in the tested datasets, the number of berries was counted manually. Hence 3DBunch's performance on accuracy can be evaluated by calculating the following indicators:

1) The R^2 value, which is influenced by the liner relationship between the actual and estimated number of berries, it provides a measure of how good the fit is between two datasets.
2) The percentage error $e_p(\%)$, which is defined as:

$$e_p(\%) = (n_e - n_a)/n_a * 100 \tag{11}$$

3) The percentage of the absolute error $e_a(\%)$, which is defined as:

$$e_a(\%) = |n_e - n_a|/n_a * 100 \tag{12}$$

4) The accuracy $\eta(\%)$, which is defined as:

$$\eta(\%) = 1 - e_a \tag{13}$$

n_e denotes the estimated berry number per bunch, whereas n_a stands for the actual berry number per bunch.

2) INDICATORS FOR COMPUTATIONAL EFFICIENCY

The computational efficiency of 3DBunch is evaluated by calculating the time taken by 3DBunch to process an image. The processing time recorded by iPad mini 4 (recorded in Table 1) to process the images from D4 and D7 (referred in Table 2) was studied. To minimize the interference of other applications or services installed on the device, the experiment was designed under the following condition:

1) The device was in flight mode.
2) The 3DBunch app was initialized without opening any other applications.

The consumed time between the determination of threshold and radius and the display of detected circles was recorded. To avoid the influence from manual operation factors, each image was analysed three times, and the average consumed time was recorded as computation time.

V. EXPERIMENTAL RESULTS AND DISCUSSION

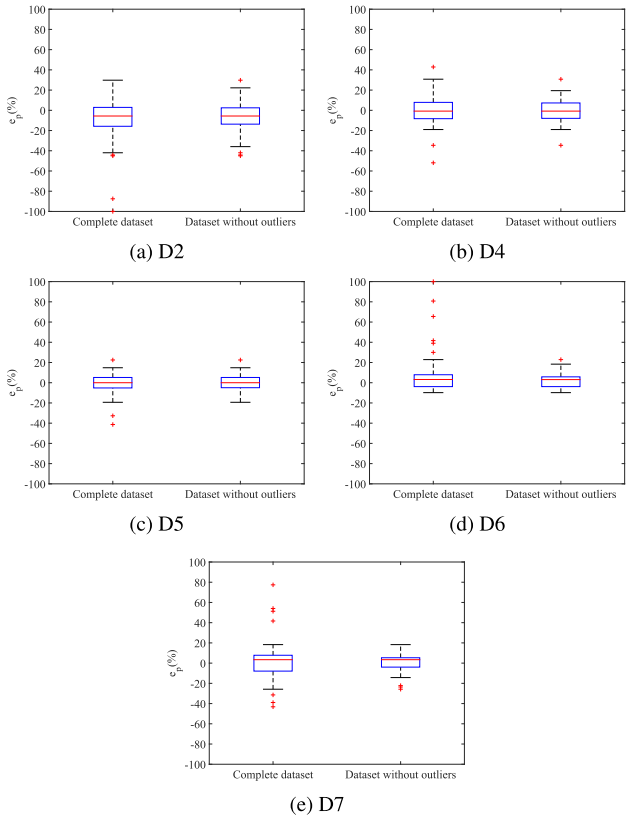
A. EVALUATION OF THE PROPOSED INTERNAL ALGORITHM IN 3DBunch

For evaluating the performance of the proposed internal algorithm regarding the accuracy, five datasets (Table 2) were tested. All related information and results are presented in Table 3 and Table 4.

The box plots in terms of error percentage are shown in Fig. 10, in which both complete dataset and dataset without outliers are considered. It is clear that distribution of median value and the second & third quantile are not shifted dramatically by those outliers, which means the capability of 3DBunch of processing images collected in the field is stable. A visual investigation of the datasets to locate the outliers was conducted. It found that the outliers (failure cases) always refer to the dry bunches (dead or half dead bunch), wrong

TABLE 4. Comparison between results with and without outliers.

Dataset	Percentage of outliers(%)	Results with outliers				Results without outliers			
		Number of images	$e_a(\%)$	$e_p(\%)$	R^2	Number of images	$e_a(\%)$	$e_p(\%)$	R^2
D2	4.11	73	8.74	-5.63	0.91	70	7.69	-5.36	0.90
D4	4.55	44	8.25	-0.79	0.92	42	7.69	-0.79	0.96
D5	3.57	56	5.20	0	0.96	54	5.03	0.81	0.98
D6	9.52	63	5.81	3.20	0.96	57	5.26	2.17	0.98
D7	12.5	56	7.84	3.40	0.77	49	6.12	3.23	0.93



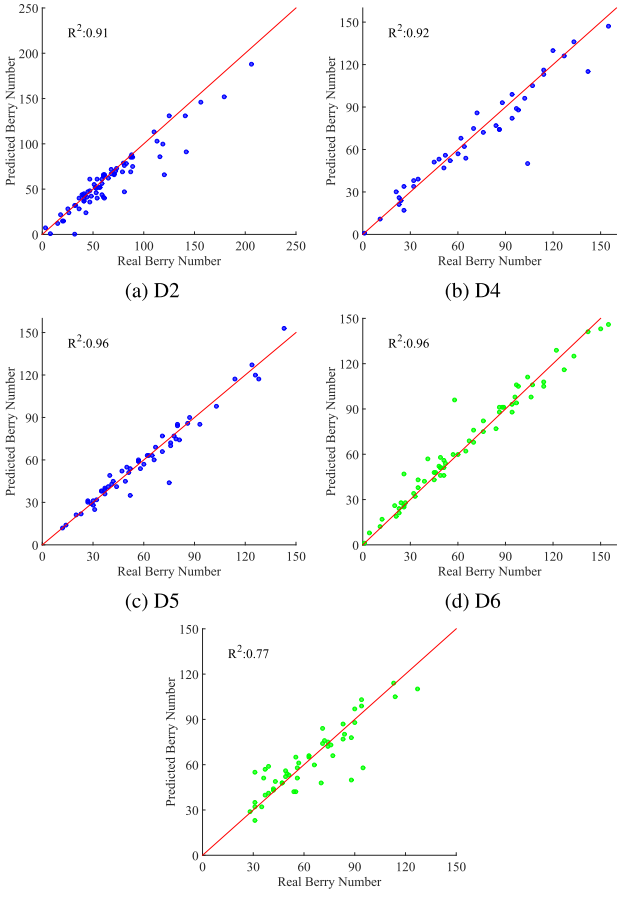
(a) D2 (b) D4 (c) D5 (d) D6 (e) D7

FIGURE 10. Box and whisker plots representing the performance of 3DBunch in term of error percentage.

(a) Dead grapes (b) Color interference (c) Noise

FIGURE 11. Failure cases.

backing board color, or noise appearing on the bunch. For the first case, the failure is caused by the big difference between actual berries and estimated berries. The dead berries were not manually counted as berries but counted as real ones by 3DBunch (see an example in Fig. 11a). As to the second case, the failure results from the bunch not being correctly



(a) D2 (b) D4 (c) D5 (d) D6 (e) D7

FIGURE 12. The quantitative relationship between the actual and estimated numbers of berries (with outliers). The red line indicates a 1:1 relationship.

segmented from the backing board. For instance, a large area of green color from surrounded canopy was reflected on the backing boards making it appear similar to that of the bunch (see Fig. 11b). The segmentation value is hard coded in the app so it fails when the color of the backing board is not highly contrasting with the bunch. For the third scenario, as shown in Fig. 11c, the outline of the bunch is extracted incorrectly, which causes the 3D model to be reconstructed incorrectly.

The datasets tested in this paper are raw data without pre-cleaning. Given that the failure cases all fell into the aforementioned three scenarios, by visually removing those "undesirable" photos, the quantitative relationship between the actual and estimated numbers of berries is presented in Fig. 12 (with outliers) and Fig. 13 (without outliers).

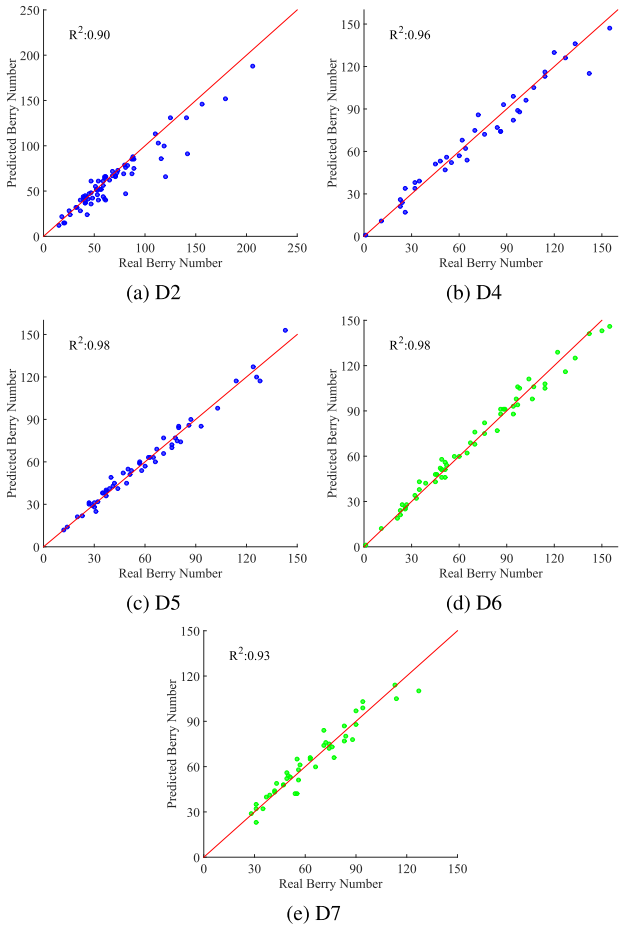


FIGURE 13. The quantitative relationship between the actual and estimated numbers of berries (without outliers). The red line indicates a 1:1 relationship.

The R^2 value, median of e_a and e_p of each dataset without outliers are listed in Table 3a. R^2 values varied from 0.86 to 0.98 (average 0.93), along with absolute error 5.2% to 8.74% (average 7.17%) and error -5.36% to -3.22% (average 0.01%), indicating the good performance of 3DBunch on both simulator and real machine. The average accuracy η of five datasets without outliers was also calculated and it provided an acceptable value of 91%.

Bunches from the tested datasets were photographed by three difference mobile devices with different resolutions (detailed in Table 2). This represents that 3DBunch is not limited to a single type of mobile device. Berry number estimation with low error in multiple conditions verifies the robustness of the internal image processing algorithm embedded in 3DBunch.

In addition, the sparsity or compactness of a grape bunch varies during its growing season. Images of datasets D4 to D7 were captured in-vivo. By referring to the modified E-L naming convention [38], their E-L stages varied among 31 (Berries pea-sized), 33 (Berries still hard and green) and 38 (Harvest). The performance of 3DBunch estimating the berry number for a bunch is promising at different E-L stages [38] according to the numbers listed in Table 3. Hence 3DBunch

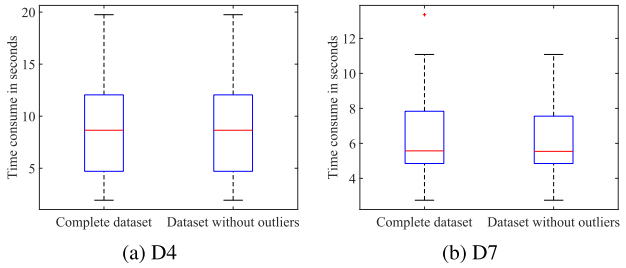


FIGURE 14. Boxplots of computational time.

(a) One single berry, Shiraz (b) Dense and compact bunch, Chardonnay

FIGURE 15. Some cases for varying computational time.

TABLE 5. Results of 3DBunch's computational efficiency study.

Dataset	Average[seconds]	Standard deviation
D4	9.12	4.91
D7	6.25	2.44

has the capability to estimate the number of berries per bunch accurately as early as 31 (Berries pea-sized), which aids the user in predicting the yield before harvest.

B. RESULTS OF THE STUDY OF 3DBunch's COMPUTATIONAL EFFICIENCY

The computational efficiency of 3DBunch which runs on iPad mini 4 (detailed in Table 2) is depicted in Fig. 14. The average computation time and the standard deviation were also calculated and listed in Table 5. The device had 9.12s and 4.91s of average time and standard deviation repectively for dataset D4. As to dataset D7, it took 6.25s and 2.44s of average time and standard deviation repectively. The outliers refer to bunches with too few or too many berries (see Fig. 15). As the computational time by 3DBunch has a linear relationship with the number of berries per bunch, considering the modest specs of iPad mini 4, 3DBunch is able to run with smooth utilization experience on most iOS devices. The low average and standard deviation processing time indicate that 3DBunch is an efficient tool comparing with conventional manual berry counting.

VI. CONCLUSION

This paper presents an innovative iOS application for fast berry counting in the field, called 3DBunch, which is now

available in the Apple App Store. It assists grape vineyard technicians, viticulturists, and grape farmers to predict grape yield in early stages by counting berries in a convenient, non-destructive and low-cost manner for both purple and green grapes based on image analysis. According to the results presented in Section V, 3DBunch is an accurate, with an average accuracy of 91%, and practical tool for end users to estimate the number of berries in varies growing stages, suitable for both *in-vivo* or *ex-vivo* conditions. The low average and standard deviation of processing time prove that 3DBunch is an efficient and robust application. The fluent app design, clear user interface with proper instructions, data retrieval & visualization and cloud data storage provide a user-friendly experience.

Further improvements of 3DBunch will include implementing multiple thresholds in grape area extraction, which will help 3DBunch extract the outline of the bunch more accurately and avoid the appearance of outliers such as Fig. 11b and Fig. 11c (c). To make it even more user-friendly, user feedback if a completely wrong image is loaded could be implemented. The speed of processing may be improved by cloud processing, but the limited internet access in field situations would not add to the user experience. A map of predicted yield variation across a block could be generated according to the location of each image taken by a smartphone. A large scale of phenotypic information of the bunch can be investigated by combining 3DBunch with existing work on flower counting [29], [39]. The 3D model reconstruction of bunch provides opportunities for obtaining more agronomic parameters of the bunch, which will be investigated in the future.

ACKNOWLEDGEMENT

The datasets used in this paper were obtained with the support of Wine Australia project DPI1401 Improved Yield Estimation for the Australian Wine Industry. All datasets are freely available for download from the Smart Robotic Viticulture group's website: http://www.robotics.unsw.edu.au/srv/datasets.html

REFERENCES

[1] Western Australia. (2020). *Australian Wine Sector at a Glance*. [Online]. Available: https://www.wineaustralia.com/market-insights/australian-wine-sector-at-a-glance

[2] Wikipedia. (2020). *Annual Growth Cycle of Grapevines*. [Online]. Available: https://en.wikipedia.org/wiki/Annual_growth_cycle_of_grapevines

[3] K. Seng, L. Ang, L. Schmidtke, and S. Rogiers, "Computer vision and machine learning for viticulture technology," *IEEE Access*, vol. 6, pp. 67494–67510, Oct. 2018.

[4] M. J. C. S. Reis, R. Morais, E. Peres, C. Pereira, O. Contente, S. Soares, A. Valente, J. Baptista, P. J. S. G. Ferreira, and J. B. Cruz, "Automatic detection of bunches of grapes in natural environment from color images," *J. Appl. Log.*, vol. 10, no. 4, pp. 285–290, Dec. 2012.

[5] D. Font, T. Pallejà, M. Tresanchez, M. Teixidó, D. Martinez, J. Moreno, and J. Palacín, "Counting red grapes in vineyards by detecting specular spherical reflection peaks in RGB images obtained at night with artificial illumination," *Comput. Electron. Agricult.*, vol. 108, pp. 105–111, Oct. 2014.

[6] D. Font, M. Tresanchez, D. Martínez, J. Moreno, E. Clotet, and J. Palacín, "Vineyard yield estimation based on the analysis of high resolution images obtained with artificial illumination at night," *Sensors*, vol. 15, no. 4, pp. 8284–8301, Apr. 2015.

[7] A. Aquino, B. Millan, M.-P. Diago, and J. Tardaguila, "Automated early yield prediction in vineyards from on-the-go image acquisition," *Comput. Electron. Agricult.*, vol. 144, pp. 26–36, Jan. 2018.

[8] S. Nuske, S. Achar, T. Bates, S. G. Narasimhan, and S. Singh, "Yield estimation in vineyards by visual grape detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 2352–2358.

[9] L. Zabawa, A. Kicherer, L. Klingbeil, A. Milioto, R. Topfer, H. Kuhlmann, and R. Roscher, "Detection of single grapevine berries in images using fully convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–9.

[10] O. Mirbod, L. Yoder, and S. Nuske, "Automated measurement of berry size in images," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 79–84, 2016.

[11] J. L. Hernández-Hernández, J. Ruiz-Hernández, G. García-Mateos, J. M. González-Esquiva, A. Ruiz-Canales, and J. M. Molina-Martínez, "A new portable application for automatic segmentation of plants in agriculture," *Agricult. Water Manage.*, vol. 183, pp. 146–157, Mar. 2017.

[12] A. Laamrani, R. Pardo, A. Berg, D. Branson, and P. Joosse, "Using a mobile device 'app' and proximal remote sensing technologies to assess soil cover fractions on agricultural fields," *Sensors*, vol. 18, no. 3, p. 708, 2018.

[13] A. Gong, J. Yu, Y. He, and Z. Qiu, "Citrus yield estimation based on images processed by an Android mobile phone," *Biosyst. Eng.*, vol. 115, no. 2, pp. 162–170, Jun. 2013.

[14] S. Cubero, F. Albert, J. M. Prats-Moltalbán, D. G. Fernández-Pacheco, J. Blasco, and N. Aleixos, "Application for the estimation of the standard citrus colour index (CCI) using image processing in mobile devices," *Biosyst. Eng.*, vol. 167, pp. 63–74, Mar. 2018.

[15] J. Qian, B. Xing, X. Wu, M. Chen, and Y. Wang, "A smartphone-based apple yield estimation application using imaging features and the ANN method in mature period," *Scientia Agricola*, vol. 75, no. 4, pp. 273–280, Aug. 2018.

[16] F. Nutini, R. Confalonieri, A. Crema, E. Movedi, L. Paleari, D. Stavrakoudis, and M. Boschetti, "An operational workflow to assess rice nutritional status based on satellite imagery and smartphone apps," *Comput. Electron. Agricult.*, vol. 154, pp. 80–92, Nov. 2018.

[17] T. Valdez-Morones, H. Perez-Espinosa, H. Avila-George, J. Oblitas, and W. Castro, "An Android app for detecting damage on tobacco (Nicotiana Tabacum L.) leaves caused by blue mold (Penospora Tabacina Adam)," in *Proc. 7th Int. Conf. Softw. Process Improvement (CIMPS)*, Oct. 2018, pp. 125–129.

[18] P. P. Singh, P. Pandey, D. Singh, S. Singh, M. S. Khan, and M. Semwal, "'Mentha Mitra'—An Android app based advisory digital tool for menthol mint farmers," *Ind. Crops Products*, vol. 144, Feb. 2020, Art. no. 112047.

[19] M. A. Carmona, F. J. Sautua, O. Pérez-Hernández, and J. I. Mandolesi, "AgroDecisor EFC: First Android app decision support tool for timing fungicide applications for management of late-season soybean diseases," *Comput. Electron. Agricult.*, vol. 144, pp. 310–313, Jan. 2018.

[20] A. C. Bartlett, A. A. Andales, M. Arabi, and T. A. Bauder, "A smartphone app to extend use of a cloud-based irrigation scheduling tool," *Comput. Electron. Agricult.*, vol. 111, pp. 127–130, Feb. 2015.

[21] D. Mbabazi, K. W. Migliaccio, J. H. Crane, C. Fraisse, L. Zotarelli, K. T. Morgan, and N. Kiggundu, "An irrigation schedule testing model for optimization of the smartirrigation avocado app," *Agricult. Water Manage.*, vol. 179, pp. 390–400, Jan. 2017.

[22] K. M. A. Vijay, K. N. C. Kumar, N. Kumar, K. Harshitha, and M. K. K. Khan, "An improved agriculture monitoring system using agri-app for better crop production," in *Proc. 3rd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2018, pp. 2413–2417.

[23] Q. Xiaohui, D. Shangfeng, H. Yaofeng, and L. Meihui, "Development and design of mobile terminal APP for greenhouse environment control," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 822–825, 2018.

[24] C.-C. Fan, R.-H. Wu, L.-L. Jau, and Y.-M. Li, "Wireless sensor and mobile application of an agriculture security system," in *Intelligent Data Analysis and its Applications*, vol. 2, J.-S. Pan, V. Snasel, E. S. Corchado, A. Abraham, and S.-L. Wang, Eds. Cham, Switzerland: Springer, 2014, pp. 77–85.

[25] L.-M. Ang, K. Seng, A. Oczkowski, A. Deloire, and L. Schmidtke, "Development of a smartphone app for berry quality assessment," in *Proc. OENOVITI Int. Netw.* Mendoza, Argentina: Vigne et Vin Publications, 2018, pp. 79–85.
[26] M. Grossetete, Y. Berthoumieu, J.-P. Da Costa, C. Germain, O. Lavialle, and G. Grenier, "A new approach on early estimation of vineyard Yield: Site specific counting of berries by using a Smartphone," in *Proc. Eur. Conf. Precis. Agricult.*, Prague, Czech, Jul. 2011, pp. 1–8.
[27] A. Aquino, M. P. Diago, B. Millán, and J. Tardáguila, "A new methodology for estimating the grapevine-berry number per cluster using image analysis," *Biosyst. Eng.*, vol. 156, pp. 80–95, Apr. 2017.
[28] A. Aquino, I. Barrio, M.-P. Diago, B. Millan, and J. Tardaguila, "VitisBerry: An Android-smartphone application to early evaluate the number of grapevine berries by means of image analysis," *Comput. Electron. Agricult.*, vol. 148, pp. 19–28, May 2018.
[29] A. Aquino, B. Millan, D. Gaston, M.-P. Diago, and J. Tardaguila, "Vitisflower: Development and testing of a novel Android-smartphone application for assessing the number of grapevine flowers per inflorescence using artificial vision techniques," *Sensors*, vol. 15, no. 9, pp. 21204–21218, 2015.
[30] L. Schmidtke, *Developing a Phone-Based Imaging Tool to Inform on Fruit Volume and Potential Optimal Harvest Time*. Bathurst, Australia: Charles Sturt Univ., 2018.
[31] M. Rahaman, M. Paul, L. Schmidtke, L. Zheng, and A. Oczkowski, "Developing a mobile App to estimate grape volume and colour for harvest," in *Proc. 17th Austral. Wine Ind. Tech. Conf. (AWITC)*, Jul. 2019. [Online]. Available: https://researchoutput.csu.edu.au/en/publications/developing-a-mobile-app-to-estimate-grape-volume-and-colour-for-h
[32] S. Liu, X. Zeng, and M. Whitty, "A vision-based robust grape berry counting algorithm for fast calibration-free bunch weight estimation in the field," *Comput. Electron. Agricult.*, vol. 173, Jun. 2020, Art. no. 105360.
[33] Wikipedia. (2020). *iOS*. [Online]. Available: https://en.wikipedia.org/wiki/IOS
[34] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. London, U.K.: Pearson, 2012.
[35] Apple Inc. (2020). *iOS*. [Online]. Available: https://developer.apple.com/documentation/glkit
[36] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[37] S. Brahmbhatt, *Practical OpenCV*. New York, NY, USA: Apress, 2013.
[38] B. G. Coombe, "Growth stages of the grapevine: Adoption of a system for identifying grapevine growth stages," *Austral. J. Grape Wine Res.*, vol. 1, no. 2, pp. 104–110, Jul. 1995.
[39] S. Liu, X. Li, H. Wu, B. Xin, J. Tang, P. R. Petrie, and M. Whitty, "A robust automated flower estimation system for grape vines," *Biosyst. Eng.*, vol. 172, pp. 110–123, Aug. 2018.

SCARLETT LIU (Member, IEEE) is currently a Senior Lecturer in vision sensing for field robotics with the School of Traffic and Transportation Engineering, Central South University, China. Her research interests include robotics with experience in robotic vision, sensing, and environment reconstruction. She does theoretical investigation on unsupervised data structure learning from the perspective of vision perception and sensing for autonomous systems, which supports decision making and autonomous system control. Her solid publication record and recognition for excellence in presentation skills and teaching confirm her reputation as an Efficient Communicator. Her experience on conducting planning and management of field work accumulated for past five years proves her capability on efficient filed operations and engagement between academic and industrial parties. Her bilingual skills have been utilized to help improve international recognition of the research team of CV-MLI and supervise international and domestic students. Her clear research plan enhances her ability to set a role model for female engineers/academics in robotics.

XIANGDONG ZENG received the bachelor's degree in engineering from Central South University, in June 2019, where he is currently pursuing the degree in transportation engineering. His research interests include computer vision and field robotics.

MARK WHITTY (Member, IEEE) received the Ph.D. degree in mechatronic engineering from the University of New South Wales (UNSW), Sydney, Australia. He is currently a Senior Lecturer with the School of Mechanical and Manufacturing Engineering, UNSW. He leads the Smart Robotic Viticulture Team, UNSW. His research interests include digital viticulture, digital agriculture, agricultural robotics, and 3D point cloud processing. He has been successful with both nationally competitive grants and engagement with industry internationally in the agricultural sector.