A Dynamic, Real-time Algorithm for Seed Counting

Mitchell L. Neilsen, Chaney Courtney,
Siddharth Amaravadi, and Zhiqiang Xiong
Department of Computer Science
Kansas State University
2184 Engineering Hall
Manhattan, KS, USA

Jesse Poland and Trevor Rife

Department of Agronomy
Kansas State University
4713 Throckmorton PSC
Manhattan, KS, USA

Abstract

Kansas State University is a world leader in the study of wheat genetics to develop new varieties that can tolerate a wide range of environmental conditions. Field Book, a new mobile application, was developed to modernize plant breeding programs around the world. Building off its success, we've developed several additional mobile apps for plant breeding. As a part of this effort, novel image analysis algorithms are being developed to model and extract plant phenotypes. This paper describes a dynamic, real-time algorithm to accurately count seeds using a modest mobile device. There are many directions for future research to enhance the algorithm's performance and accuracy. The new algorithm can be used to count a wide variety of different types of crop seeds for high-throughput phenotyping.

Keywords: Dynamic real-time algorithm, high-throughput phenotyping, image processing, mobile applications.

1. Introduction

A new project at Kansas State University, called BreadPheno, seeks to leverage novel advances in image processing and machine vision to deliver transformative mobile applications through several existing breeder networks. Building on the success of Field Book, user-friendly mobile apps for field-based, high-throughput phenotyping (HTP) are being built and deployed for wide distribution (http://www.wheatgenetics.org/phenoapps). Novel image analysis algorithms are deployed to model and extract plant phenotypes. A robust development pipeline is assisted by both real-time field testing by breeding collaborators and the broader community using the apps to explore plant growth and quantitative differences under genetic control. A diverse set of crops and breeder networks are engaged to provide us with a diverse set of target plant phenotypes, environments, breeding programs, and working cultures.

Dramatic increases in the speed and ability to collect precision phenotypic data are needed to decipher plant genomes and accelerate plant breeding. Over the past decade, the availability of genomic data has exploded while the methods to collect phenotypes have made minimal advancements. This has resulted in a dramatic imbalance in data sets connecting genotype to phenotype and highlights phenotyping as the remaining major bottleneck in plant breeding programs. This project seeks to advance the field of graphics and modeling, data mining and deep learning through integration of simultaneous ground-truth phenotypic measurements and imaging with mobile technology. By combining data from multiple research programs with ground-truth breeder knowledge, this project will lay the foundation for collecting training set data that can subsequently be used to extract and quantify complex phenotypes using advanced algorithms.

Food and nutritional security is a grand challenge in the coming decades. The global population will increase to over 9 billion and food demand will grow by more than 50%. To address this challenge, novel advancements to leverage genomic information and expedite improvement of plant varieties are needed. While genomic information has become inexpensive and readily available, the complementary phenotypes needed to understand the function of plant genomes and make selections in breeding programs has remained static, with little development, particularly for phenotypes collected from field trials in breeding programs.

By focusing on novel algorithms delivered through mobile apps, innovative phenotyping tools can be rapidly deployed through readily available and highly penetrant mobile technology. This approach will enable rapid dissemination and broad usability. Collectively equipping thousands of breeders around the world with tools for rapid collection, processing and analysis of complex phenotypes will provide the foundation for increasing genetic gain that will ultimately result in improved productivity, food security, nutrition and income of smallholder farmers and their families in developing countries.

Watershed segmentation algorithms can be used to automatically separate particles in an image that touch. The algorithm calculates the floating point Euclidean distance map (EDM) of each pixel in the image, where Euclidean distance is defined in the usual way as the closest Euclidean distance to any background pixel [2]. Then, it finds the ultimate eroded points (UEPs) which are the peaks or locally maximal connected components of the EDM. It then dilates each of the UEPs as far as possible, either until the edge of the particle (the background) is reached, or the edge

978-1-943436-09-5 / copyright ISCA, SEDE 2017
October 2-4, 2017, San Diego, California, USA

touches a region of another growing UEP region. In the classical algorithm, the regions are flooded in different directions on each step of the algorithm. In previous work, we extended watershed segmentation algorithms for high-throughput phenotyping using static images of seeds [3].

This paper presents a new dynamic, real-time algorithm to compute the number of seeds flowing across a back-lit translucent screen captured as a movie or in real-time on a modest mobile device. The goal is to track seeds between frames to consistently count seeds within 1% of the actual count. The new algorithm leverages the fact that the seeds flow from top to bottom to more accurately count the seeds.

Section 2 describes the current state of the art, and Section 3 describes our new algorithm. Section 4 provides a brief analysis. Finally, Section 5 concludes the paper.

2. State of the Art

Rapid collection and analysis of large amounts of phenotypic data is known as high-throughput phenotyping (HTP). Image-based methods for computing phenotypic measurements provide a promising solution for high-throughput systems in plants. There have been many successful applications of image analysis to plant phenomics. However, these applications of image-based HTP have been largely concentrated on laboratory or controlled environment facilities. While providing highly accurate phenotypes, the ability to transfer these experiments to field conditions remains limited. Moreover, the limited capacity and infrastructure needed to develop controlled environment HTP platforms makes this approach unpromising for breeding programs that currently manage population sizes much larger than even the world's largest facilities.

Field-based phenotyping remains the major bottleneck in most breeding and genetics programs, including cassava and wheat. As recognized by the breeders, this bottleneck can be overcome by adopting and/or developing innovative, high-throughput, and accurate hand-held HTP tools that can greatly increase the ability of breeders to generate useful data more cheaply and rapidly. As such, there is considerable willingness and motivation in the breeding community to test and adopt new technologies that can make the breeding program more efficient.

We have developed and deployed several user-friendly mobile apps that have been adopted by hundreds of breeding programs around the world. They are available online at no cost. Our apps have been designed to be standalone, simple, intuitive, and effective. They are delivering a significant increase in breeding productivity and are the foundation for the current project. These tools focus on simplifying and digitizing data collection by focusing on procedures that are common to all breeding programs. Current development is complete and a beta version has been released for '1KK', an app that can give accurate size and shape parameters for seeds and tubers/roots. With the namesake of 1KK for capturing a thousand (1K) kernel (K) traits, the app extends to any size and scale of 'seeds' (Figure 1). The app combines the image analysis library OpenCV [4, 5] with a USB scale to capture both an image of a collection of seeds and a weight measurement. Seeds or roots are spread on a colored mat that includes reference circles of a known size to convert pixels to absolute size in mm.

Integrated image processing scripts identify individual seeds/tubers/roots to calculate the length and width while attempting to count the total number. Average weight (thousand kernel weight) is calculated by dividing the total weight determined by the scale. Raw data are stored within an internal database and parameter averages are also calculated and stored on a per sample basis. Data are exported as a flat file and can be uploaded to a database.

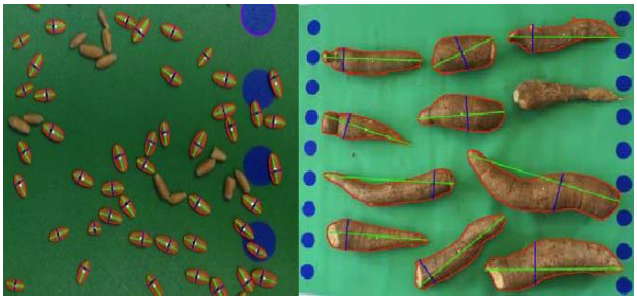

Figure 1: Wheat and cassava roots in 1KK app.

Figure 1 shows an example of the current 1KK app screen after capturing an image of wheat on the left and cassava roots on the right.

3. Dynamic, Real-time Algorithm

In addition to size, it is also important to accurately count the number of seeds. One approach is to place the seeds on a fixed background, as shown in Figure 1. This only works well for a relatively small number of seeds.

Figure 2: Lab test sled

Another approach is to capture the seeds as they flow across a simple backlit translucent sheet as shown in Figure 2. The algorithm needs to be able to process images

in real-time for two different use cases. In the first case, the user needs to count the number of seeds in a packet. In the second use case, the user needs to fill up a pack with a fixed number of seeds. For this paper, we only focus on the first use case, but we have designed the algorithm to be fast enough to give users feedback in real-time so that the application can display the current number of seeds in the packet and let the user know when the packet is getting almost full.

Empirically, we have determined that the lowest capture rate is 60 frames/second (fps) for practical use. This is a standard capture rate for mobile devices. For more accuracy, a higher capture rate, such as 120 fps, using the typical slow-motion settings can be used. The algorithm relies on libraries from OpenCV [4, 5]. For each frame:

1. Convert the frame image to grayscale.
2. Reduce noise using a small 5x5 Gaussian blur.
3. Compute the absolute difference between the current frame and the first frame (the background).
4. Apply a binary threshold to the absolute difference (default threshold is 50), below 50, set to black; for pixels above 50, set to white.
5. Further reduce noise using morphological opening.
6. Find image contours using a binary threshold. Adaptive mean thresholding and adaptive Gaussian thresholding work equally well.
7. Predict the current location of each existing seed based on its previous location and the *average flow rate*. If the *average flow rate* is not known yet, just use 0 or 1; that is, consider the predicted location to be initially the same or near the previous location.
8. Process contours of a reasonable size from bottom to top. Contours are considered to be a reasonable size if they are at least one-fourth the average contour size. If the *average contour size* is not known, then an initial value based on previous runs is used. For testing we used different values for the initial value, even down to 200. This has little impact on the algorithm. For each contour:
    a. Determine which unmarked seed's predicted location is closest to the centroid of the current contour.
    b. If no seed is within the *maximum seed distance*, which is computed as 4**average flow rate*, mark the contour as a new seed, and add the new seed to the list. If the *average flow rate* has not been computed yet, just use a default *average flow rate* based on the average of previous runs. We used an initial default of 34.
    c. Otherwise, update the coordinates of the closest seed using the centroid of the contour as the new location and mark the seed as visited. Increment the number of times the seed has been updated; i.e., associated with a contour, and reset its' age back to zero.
9. For each seed not marked, age the seed. If the seed is aged for more than half of the screen height and the *age* is more than the number of *updates*, then remove the seed from the count. The number of times a seed should appear on some frame, while it is passing from top to bottom, is computed as the height of the frame, say 720, divided by the *average flow rate*; e.g., 34, to give 21.17. Then, the maximum age is set to *int*(21.17/2)+1 = 11. The *maximum seed age* is computed dynamically based on the *average flow rate*. Since the *average flow rate* is not known initially, a static value based on the default flow rate is used initially.
10. Compute the *average flow rate* for all active seeds (in pixels per frame) and the *average seed size* (in pixels). Note that the perceived size of a seed may change as it travels from top to bottom, so we use all samples in our average. Dynamically update the values for *maximum seed distance* and *maximum seed age* based on the *average flow rate*.

For example, consider frame 384, shown in Figure 3. The first step is to convert the frame to grayscale. Then, reduce some noise using a Gaussian blur with a small 5x5 kernel as shown in Figure 4.

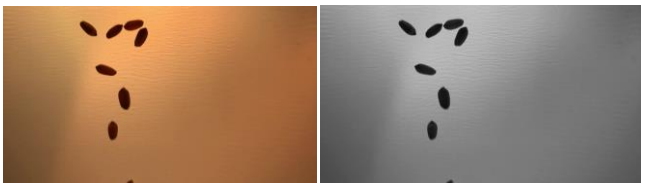

Figure 3: Captured wheat image converted to grayscale

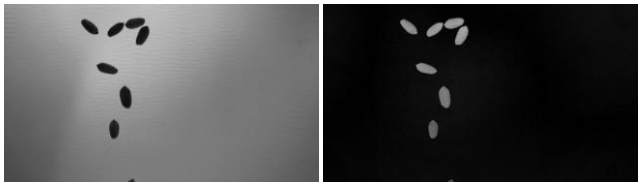

Figure 4: Gaussian blur and absolute difference

The background is just the first frame with no seeds. To remove the background, compute the absolute difference between the current frame and background as shown in Figure 4. Apply a binary threshold to the result with a threshold around 50. The background is set to black (0) and the foreground to white (255) as shown below in Figure 5.

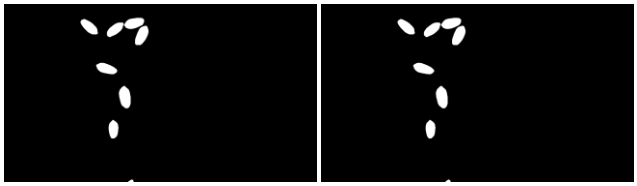

Figure 5: Binary threshold and morphological opening

An optional step is to apply morphological noise removal by using an opening which is the dilation of the erosion of a given set. Closing can also be used to remove noise, but it tends to cause individual seeds to merge or cluster together which we are trying to avoid. Finally, find the contours using the traditional border following technique by Suzuki and Abe [6], watershed segmentation by Soille and Vincent [7], or extensions to watershed segmentation by Neilsen, et al. [3]. Figure 6 shows the resulting contours in blue.

Figure 6: All seeds matched to contours (frame 384)

The contours ranged in size from 808 to 7272 pixels. Contours are considered to be a reasonable size if they are at least one fourth the average contour size; e.g., 202. We started with an upper bound as well, but found that it is not necessary because no contours were ever found to be larger than our upper bound.

Unlike previous approaches to tracking which count objects as they cross a line [9], our approach is to track seeds for their entire trip across the visible screen. To determine which seed corresponds to a given contour, we first try to predict where the seeds will be in the current frame based on their location in the previous frame and the current average flow rate which is measured in pixels per frame. Since the seeds generally flow from top to bottom, the predicted location is simply computed as:

predicted x = prior x
predicted y = prior y + average flow rate

Then, we determine which seed's predicted location is closest to the current contour's centroid. Empirically, we determined that 4 * *average flow rate* is a reasonable bound for the maximum distance that a seed can be away from the contour and still be considered as a seed represented by the contour. The coefficient 4 can vary. If we use 2, then a total of 220 seeds are created in this example, but the resulting count is still correct at 217 because 3 of the seeds will be removed during analysis.

If no seed is found within that distance, the contour is marked as a new seed, and its age is set to zero. If a seed is found, then the current coordinates of the seed are updated and the seed is marked as found and its age is set back to zero and the update count is incremented by one. If a seed ages for more than the maximum age which is computed as the number of frames in at least half of the screen, it is considered lost and removed from the count. Such zombie seeds typically result when several seeds merge and split simultaneously in different nearby contours and a low coefficient, such as 2, is used. After each frame is processed, the average flow rate and average contour size is computed and displayed, along with the current frame number and seed count. In this example, in the very next frame, seeds 52, 53, and 55 collide with each other; they are represented by a single contour with centroid at point 55 in Figure 7.

Figure 7: Three seeds in single contour (frame 385)

The predicted locations for seeds are shown as little blue dots labeled with white numbers in Figure 7. In addition, seeds 52 and 53 are aged by one. Since the images are 1280x720, if they make it to an age of 12 at the current flow rate 33, they will be discarded and the seed count will be decremented. In the next frame, seed 53 separates from the group, so its age is set back to 0. The location of 52 is predicted as shown below in Figure 8.

Figure 8: Two seeds in single contour (frame 386)

After a few more frames, all three seeds are separated and assigned to their own contours as shown in Figure 9.

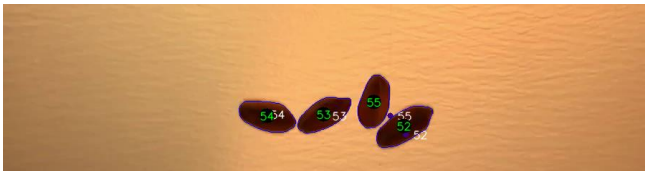

Figure 9: Seeds separated (frame 389)

The final seed count is exactly correct, 217 seeds. The same experiment was conducted twenty times with an error of at most one seed. In the two erroneous cases, the algorithm underestimated the number of seeds by one; e.g., 216 instead of 217, thus, achieving the goal of an error rate of at most 1% for this set of seeds.

Figure 10: Final seed (frame 837)

Since the final seed to be counted is number 217, and the final count is 217, none of the seeds created were removed. The same algorithm can be used with different backgrounds to count a wide range of different seed types including corn, as shown in Figure 11.

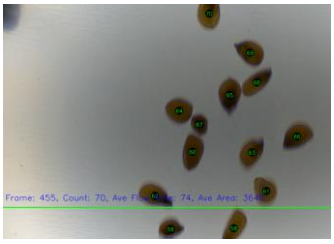

Figure 11: Corn test

For this test, a total of 227 corn seeds are counted. For a bigger challenge, we also used the counter on much smaller canola seeds as shown below in Figure 12.

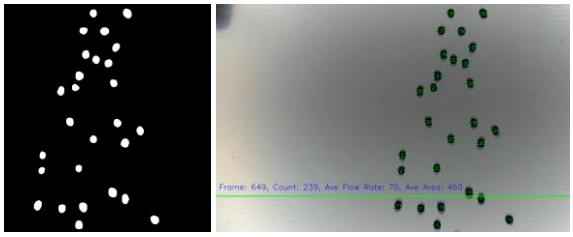

Figure 12: Canola test

Note that the flow rate tends to be higher for smaller seeds. Due to the small seed size, there was more merging and splitting as well. For this test, the actual seed count was 547, and all twenty runs resulted in counts between 545 and 547, again achieving the desired precision.

4. Analysis

The algorithm was analyzed by using a standard glass sled set at a 15 to 20 degree angle, as shown in Figure 2, to allow seeds to flow slowly down the sled with flow rates of 32 to 88 pixels per frame and video capture rates of 60 to 120 frames per second using standard and slow motion video settings. Thus, a typical run took about 4 to 5 seconds to deliver 200 to 300 seeds using the set up shown in Figure 2.

We tested different translucent materials for the background including plastic chemical jugs, milk jugs, cabinet liner material, etc. Clear glass does not perform as well, but the other translucent materials perform equally well as shown in Figure 13.

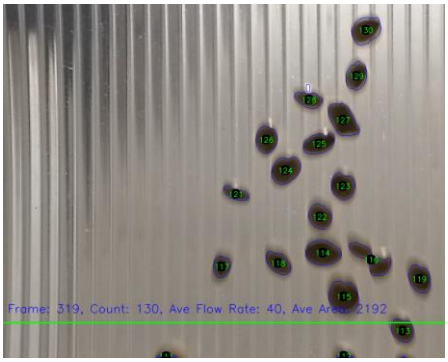

Figure 13: Test with ribbed cabinet liner material

To the extent possible, we want to have an algorithm that can adapt to different types of seeds and different capture devices. We determined that the most influential factors in controlling the error rate where:

1. The rate at which the seeds were delivered on the sled. We are developing a mechanical hopper to deliver the seeds at a fixed rate using a stepper motor to control the seed delivery rate. Currently, we rely on the steady hand of a graduate student. We delivered seeds at roughly 50 seeds per second. Seeds delivered at more than 100 seeds per second resulted in more errors.
2. The rate at which frames were captured by the camera -- most of our testing was done using 60 or 120 frames per second. The error rate jumped dramatically if we dropped below 30 frames per second as shown below in Figure 14.
3. The image resolution used to capture the frames. The highest resolution we tested was the standard 1280 x 720 pixels. With a scale of 2/3 in both the x- and y-axis; e.g., a resolution of 853 x 480 pixels, we were able to bound the error by 1%, but below that, the error rate exploded as shown below in Figure 14.

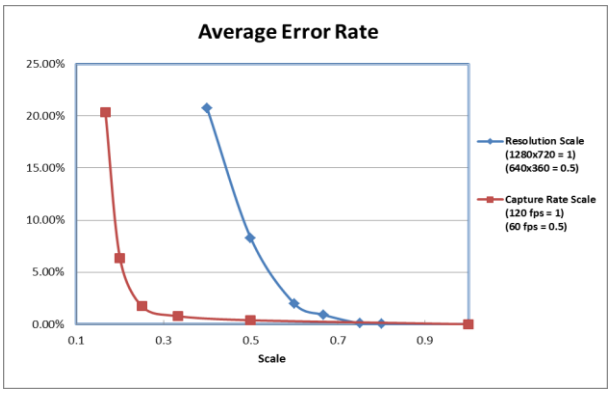

Figure 14: Observed error rates

5. Conclusions and Future Work

Building on the success of Field Book, a collection of user-friendly mobile apps for field-based, high-throughput phenotyping (HTP) are being developed and deployed. As a part of this effort, novel image analysis algorithms are being developed to model and extract plant phenotypes. This paper described a simple, new approach to accurately count seeds flowing down a backlit plane using a modest mobile device.

The algorithm provides a simple, data driven and dynamic mechanism for fine tuning the algorithm's parameters based on the average contour size and average flow rate detected as the algorithm processes each frame. We also tested the algorithm to show that it is robust for use with many different translucent background materials. By testing with a diverse set of seeds, and counting each set of seeds for twenty trials, we were able to demonstrate high confidence in the counting results. We plan to conduct a more thorough statistical analysis to identify exactly which parameters significantly influence the algorithm's performance, precision, and accuracy. Clearly, the flow rate is an important factor, we plan to design a mechanical feeder and hopper to accurately control the flow rate. The maximum flow rate is still comparable to commercial solutions costing thousands of dollars more.

There are several directions for additional work. We have extended the app to process live video streams in real-time. Based on our preliminary results, the processing overhead imposed by the algorithm is low enough to allow both the recording and processing of the video at the same time. Also, it may not be important to keep the video if the user only needs a count. The user will have the option to save the video for additional processing.

The existing algorithm could be extended by considering contour size and applying our extensions to the watershed segmentation algorithm to go ahead and split contours that actually represent more than one seed [3]. However, the cost of this additional processing may be prohibitive. Preliminary observations have shown that the seeds may not cluster long enough during flow to warrant the cost of the additional processing. We could also simply associate more than one seed with a given contour if it is sufficiently large and the seeds are sufficiently close.

All of the code is designed to run in both desktop and mobile environments. This facilitates the design of lesson plans for use in the K-12 classroom [8].

We plan to 3D print simple sleds backlit with LEDs and camera mounts with add-on hoppers and mechanical devices to control flow controlled by a stepper motor.

More sophisticated metrics could be used to predict the seed location or determine the seed which is "closest" to a given contour; e.g., we could use the Mahalanobis distance or a probabilistic metric based on where a seed is likely to fall. However, it may be the case that the simple Euclidean distance is good enough to obtain the desired results in a timely fashion. The guiding principle that we have employed in developing this application is to keep it simple. This has enabled us to deploy an efficient app on mobile devices which can efficiently count seeds in real-time.

Acknowledgements

We appreciate the many fine comments received from the reviewers, and have incorporated all of their feedback into our final paper.

This material is based upon work supported by the National Science Foundation under NSF-BREAD Grant No. 1543958. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] T.W. Rife and J. A. Poland, "Field Book: An open-source application for field data collection on Android", In *Crop Science* **54**, 1624-1627, 2014. DOI: 10.2135/cropsci2013.08.0579.

[2] F. Leymarie and M. D. Levine, "Fast raster scan distance propagation on the discrete rectangular lattice", *CVGIP: Image Understanding*, pp. 84-94, 1992. DOI:10.1016/1049-9660(92)90008-Q.

[3] M.L. Neilsen, S.D. Gangadhara, T. Rife, "Extending watershed segmentation algorithms for high throughput phenotyping", in *Proceedings of the 29th International Conference on Computer Applications in Industry and Engineering*, Denver, CO, Sept. 26-28, 2016.

[4] G. Bradski, "The OpenCV library", *Dr.Dobb's Journal*, *25*(11), 120-125, 2000.

[5] Itseez, "Open Source Computer Vision Library", 2016. Retrieved from https://github.com/itseez/opencv. Homepage: http://opencv.org.

[6] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following", CVGIP 30 1, pp. 32-46, 1985.

[7] P. Soille and L.M. Vincent, "Determining watersheds in digital pictures via flooding simulations", Lausanne-DL tentative, International Society for Optics and Photonics, 1990.

[8] M.L. Neilsen, J. Shaffer, and N. Johnson, "Time-lapse photography for K-12 education", In Proc. of the 11th International Conference on Frontiers in Education: Computer Science and Computer Engineering, July 27-30, 2015.

[9] O. Javed, M. Shah, and A. Yilmaz, "Object tracking: A survey", ACM Computing Surveys, 38:13, 2006.

[10] P.C. Mahalanobis, "On the generalized distance in statistics", in Proceedings of the National Institute of Sciences in India, Vol. 2, No. 1, pp 44-55, 1936.