

Analysis of food processing crystal images

Qiyue Liang¹, Ali Shakouri¹, and Jan P. Allebach¹

¹Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907

Abstract

In this paper, we study a food ingredient processing that has a total of ten fabrication stages, where the first stage is known as crystal seed count. In this stage, manufacturers study under a microscope the food sample and count crystal seeds per unit area. Sample preparation has imperfections (e.g. air bubbles) that need to be disregarded. The number of seed crystals per unit area is a key indicator for the quality of subsequent processing steps. This paper proposes a method for automating the crystal counting and air bubble removal processes in the seed count stage, in order to save manufacturers considerable time and money. An automated image processing pipeline for crystal counting and air bubble removal is employed in the proposed method. In addition to the proposed pipeline for crystal counting and air bubble removal automation, we also introduce an interactive GUI which is operated based on this pipeline. This GUI can not only be used by manufacturers of the particular crystals that we examine in this paper, but can also benefit other manufacturers in the food industry that face similar issues. By utilizing this automatic crystal counting pipeline and GUI, food crystal fabrication can be made more efficient. This paper contributes to the field of food processing by presenting a practical and innovative solution to a common challenge in the industry.

Introduction

With an ever-increasing demand for safe and high-quality food products, the food industry is a vital component of the global economy. Food manufacturers have always been exploring ways to improve their products' quality and consistency. Crystal formation is involved in a variety of food products. Food ingredient processing is a complex process that usually involves a number of steps, including seed crystal nucleation, growth, and harvesting. To ensure that uniform crystal structures are formed and the final products meet standard requirements, each step must be carefully controlled. There are ten steps in the food processing we are discussing in this paper, but we will only be focusing on the seed counting stage, which is the first manufacturing step. In this step, manufacturers are counting crystal seeds, neglecting any air bubbles that are results of sample preparation on a glass slide under a microscope. Traditionally, these crystals have been counted manually, which is time consuming and error-prone. Due to differences in perception and interpretation, manual counting may also result in variations in counting accuracy. Computer vision algorithms have made automated crystal counting feasible and efficient.

A complete pipeline is presented in this paper for pre-processing microscopic images of crystals, analyzing them automatically to detect and remove air bubbles, count crystals, and determine other related information such as crystal size and what

percentage of each microscopic picture is covered by crystals, which are important to manufacturers. All of the steps involved in the pre-processing pipeline fall under traditional computer vision techniques, such as edge detection, morphological operations, and flood filling. A special feature we use during the pre-processing stage is adaptive thresholding, which is a thresholding technique used to deal with uneven backgrounds. The pre-processing section of this paper will discuss in more detail what parameters we choose for each step of the preprocessing stage, explain the mathematical and theoretical motivations, and show sample results we obtain for each step.

We will discuss our crystal separation technique following the pre-processing section, which is one of the most important new ideas proposed in this paper. The crystal separation is crucial since pre-processing the pictures simply removes the background and unrelated information. However, the crystals are not separated into single ones. They may form a large cluster connected to one another. We propose a crystal searching and separation algorithm referred to as "four-pointers search" in this paper. We will discuss the details of this algorithm in the section called "Crystal Separation". Using the separation algorithm, we should be able to locate each crystal and obtain information about its size.

The crystal analysis results can also be used to obtain other important information that is useful to the manufacturers, such as the percentage of the image occupied by crystals. As the users of this algorithm are manufacturers in the industry, we have designed a graphical user interface to make it easier for them to use our algorithm for automatic crystal analysis. Toward the end of this paper, we will discuss potential future directions for this project, which are largely based on deep learning.

Related Works

Our crystal detection pipeline begins with preprocessing, where adaptive thresholding is used to deal with non-uniform background. Before we implement any algorithms to deal with the crystals, we must first remove the air bubbles as required by the manufacturer. We first apply a bilateral filter [1] to enhance the picture and then use Hough circle to locate the bubbles. Hough circle is a special case of the Hough transform [2] algorithm. After identifying the bubbles, we cover them with the background color, which has the same effect as removing them.

After we remove the bubbles we apply adaptive thresholding on the post bubble removal picture. Adaptive thresholding uses local information to classify all pixels in the image as either foreground or background. The concept of adaptive thresholding was first introduced in [3], combining Gaussian smoothing with local thresholding to detect nuclei in noisy and cluttered microscopic images. Another very common method of adaptive thresholding is adaptive mean thresholding, which can be found in

[4]. Adaptive mean thresholding uses local mean instead of Gaussian weights. For our specific task, we discovered that adaptive Gaussian thresholding works better than adaptive mean thresholding. Hence, we are using adaptive Gaussian thresholding in our pipeline, and all of the adaptive thresholdings referred to in this paper are adaptive Gaussian thresholdings.

Adaptive thresholding increases the clarity of edges along crystals for an unevenly illuminated and fussy image such as ours, but it also generates additional noise due to a limited view of the local area. In order to remove the noise, we generate a mask for each microscopic image, using computer vision techniques such as Canny edge detection [5], morphological opening and closing [6], and flood filling. This set of pipelines, including image segmentation, thresholding, and other machine vision strategies, has been used by researchers in the field of histopathology for a long time. Under a microscope, tissue sections are observed by a pathologist's naked eye, and the visual information is analyzed manually using their priori medical knowledge.

Although our manufacturer's purpose is to analyze the fabrication of the crystal rather than the structure of the cell, the image processing techniques employed in both tasks are broadly similar. In review papers such as [7] and [8], these pipelines have been summarized. In general, image processing pipelines for black and white microscopic images can be classified as follows: thresholding based, edge based, and region based. The thresholding approaches [9], [10], [11] usually involves nonlinear contrast enhancement, the application of a threshold either globally or locally, and morphological operations such as erosion and dilation. Edge-based approaches [12], [13], [14] also utilize morphological operations and other image processing techniques, such as contrast enhancement or smoothing. Unlike thresholding, edge-based approaches use edge detection techniques to enhance the image rather than thresholding. Common edge detectors include Laplacian [15], Canny [5], and Sobel [16] edge detectors. Region-based approach [17], [18] usually identify starting points based on morphological operations and then apply segmentation like the watershed approach [19] to locate clusters of crystals. In our approach, we combine thresholding and edge-based approaches. Adaptive thresholding is used to locate the crystal clusters and edges are used to refine thresholding results.

Our paper focuses on the initial stage seed count of the crystal fabrication, and our algorithm for separating crystals does not require the use of deep learning. We decide to use deep learning to deal with future stages, which have more complicated crystal structures. The deep learning approach [20], [21] usually trains a model either supervised or unsupervised to segment crystals or count crystal density, instead of using algorithms to fulfil the segmentation. Our idea of deep learning to automatically identify crystals involves two steps. The first step is to generate pseudo-masks with CycleGAN [22] for training a supervised segmentation model later on. This is due to the fact that we do not have any ground truth masks from the manufacturer, we only have crystal counts. In the second step, we identify the crystals using mask R-CNN [23]. The masks used for training mask R-CNN are obtained from the earlier mask generation step.

Background and tasks

It is important to have an understanding of the background and the objectives of this project in order to better understand the

reasons for the design of our pipeline. The manufacturer we are working with prefers to remain anonymous, so we will not identify it in this paper. The complete crystal fabrication process used by the manufacturer comprising a total of ten steps. All of our algorithms in this paper focus on the first step the seed count stage. The manufacturer believes the final outcome of the food processing is related to the number of seed crystals per unit area in the first stage. Thus, it is important to identify potential problems at an early stage so that the manufacturer can save time and money. An example seed count stage image is shown in Figure 1 (a).

The manufacturer has a machine that can generate an automatic count data, as shown in Figure 1 (c). However, that data is not used by the manufacturer because they don't trust it. The crystals are counted manually by the manufacturers by averaging the number of crystals in four regions as shown by yellow boxes in Figure 1 (b). The size of each window in the picture is 300×300 pixels. There are also solid black bubbles visible in the photographs. These bubbles are air bubbles which result from sample preparation between two microscope slides. They are not crystals and should be detected and disregarded when counting the crystals. It would therefore be helpful if we can develop programs that could count crystals automatically, generate the size and shape distribution histograms, as well as detect bubbles. The final goal for the future is to evaluate images from earlier stages and determine whether or not the resulting crystal fabrication will be successful. If the algorithm suggests that the final results are not satisfactory, the manufacturer could halt the process at an earlier stage and save money. This final goal will be likely achieved by training a decision making deep learning model, which is outside the scope of this paper.

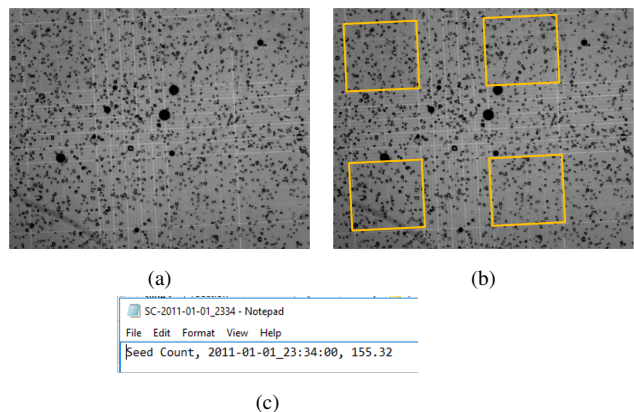


Figure 1: (a) An example of seed count stage image. (b) Regions in the seed count stage image where the crystals are manually counted by the manufacturer. (c) An example of the machine count of the seed image, which is discarded by the manufacturer.

Pre-processing Bubble Detection and Removal

Our crystal detection algorithm can be divided into two steps, pre-processing and crystal separation. The first step of pre-processing is air bubble detection and removal. The air bubbles are dark and solid, and we are using the Hough circle algorithm [2] to identify them. It is easier to explain how this algorithm works in terms of the linear case. Suppose there is a straight line

as shown in Figure 2 (a). In polar coordinates, a straight line can be expressed with Equation 1. By selecting a specific pair of x and y from the line, we can plot r versus Θ , which has a sinusoidal shape. By selecting more pairs of x and y 's, we can plot more of these sinusoidal curves, as shown in Figure 2 (b). These lines intersect at one point, which is the r and θ of the straight line we would like to detect. Using this method, we can identify potential straight line locations in the image, the more curves intersecting, the greater the likelihood that a straight line exists in the current location where we are searching.

The searching process is similar for circle detection, except that it requires three parameters instead of two to define a circle. The three parameters are: x coordinate of the circle center, y coordinate of the circle center and radius. In order to improve the detection results, we can set up extra parameters to limit the detection, thereby increasing the recall rate and decreasing false alarms. During the search, we specify six additional parameters, dp , $minDist$, $param1$, $param2$, $minRadius$, and $maxRadius$. The parameter dp is the inverse ratio of the accumulator resolution to the image resolution. The accumulator is an array that accumulates the votes corresponding to a particular circle. A smaller value of dp corresponds to a higher resolution of the accumulator array, which can lead to better detection of circles with smaller radii, but may also lead to more false detection. $minDist$ is the minimum distance (in terms of pixels) between the centers of the detected circles. $param1$ is the higher threshold of the two passed to the Canny edge detector. $param2$ is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. The parameters $minRadius$ and $maxRadius$ specify the minimum and maximum radius (in terms of pixels) of the circle that may be detected. During the search process, we set the default value of dp to 1.2, $minDist$ to 3 pixels, $param1$ to 15, $param2$ to 30, $minRadius$ to 3 and $maxRadius$ to 100. By setting these parameter values, we are able to reject many circles that are falsely detected.

As all of the air bubbles are dark and solid black, we can set an additional gray level limit and select from the candidates from the previous step. We will only keep the circles with an average gray level less than 30, since the air bubbles are darker than crystals. If the candidate circle is too light, it is not likely that it is an air bubble. To increase the accuracy of the Hough transform algorithm, it is beneficial to blur the background. In this study, a bilateral filter is applied prior to conducting the Hough circle algorithm. Figure 3 shows an example of all of the bubbles detected with the Hough transform algorithm, and the detected bubbles are indicated using green circles. After the bubbles are detected using Hough transform algorithm, they are filled with background color derived from the average of the 100×100 surrounding pixels (the center of this 100×100 area is the same as the center of the detected bubble).

$$r = x \cos \Theta + y \sin \Theta \quad (1)$$

Pre-processing Pipeline Adaptive Thresholding

Once the bubbles are detected and removed, we apply a pre-processing pipeline to the image. The entire pre-processing

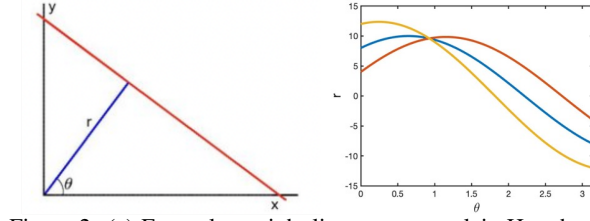


Figure 2: (a) Example straight line we use explain Hough transform algorithm. (b) It is the point where all of the curves intersect that determines the values of r and θ that can be used to calculate the polar coordinates for the straight line we want to detect.

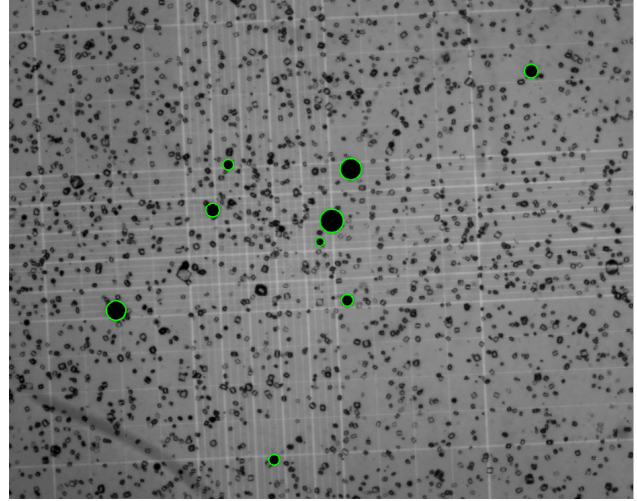


Figure 3: (a) Results of bubble detection using the Hough transform algorithm, with the detected bubbles circled in green.

pipeline is shown in Figure 4. The main idea is to use adaptive thresholding instead of a global threshold like Otsu's method [24] to separate the foreground and background. This is because of the uneven illumination on the background of the images. And since adaptive thresholding generates some noise in the background, we can use a mask to remove the noise. The mask generation pipeline is shown at the top of Figure 4, and the adaptive thresholding pipeline is on the bottom. Figure 5 explains why adaptive thresholding is used instead of a global threshold like Otsu's method. Since Otsu's method uses a threshold for the entire image, and the images might be unevenly illuminated, Otsu's method may not be able to separate all the crystals, as can be seen on the bottom left of Figure 5 (b).

Otsu's thresholding algorithm is shown in Equation 5, where T stands for the global threshold, $dst(x,y)$ is the value of the current pixel after being thresholded, $src(x,y)$ is the gray level of the current pixel before being thresholded, $maxValue$ is the maximum value of the gray level scale we choose, usually 1 or 255 depending on the image system. Basically, using Otsu's method, we check whether the current pixel's gray level exceeds the global threshold, which is selected based on the histogram distribution of the entire image. We set the gray level of a pixel to the maximum gray level if it exceeds that threshold, otherwise if its gray level is lower than threshold, we set it to zero. While this method works well for images with even backgrounds, it is not the case for our microscopic images.

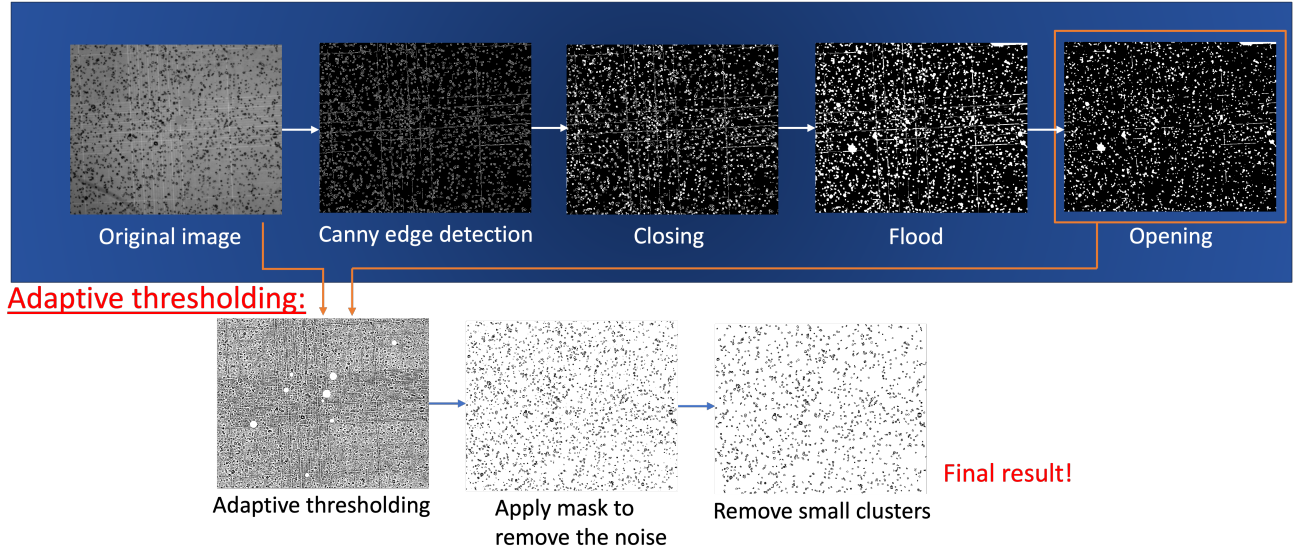


Figure 4: Pre-processing pipeline.

To better capture the local illumination information, we choose the adaptive Gaussian thresholding algorithm. The threshold of adaptive Gaussian thresholding is determined by a Gaussian-weighted sum of neighborhood values minus a constant. For adaptive Gaussian thresholding, the size of the window and the constant choice are crucial factors. As shown in Figure 6, a larger constant helps to reduce the background noise (this is the equivalent of lowering the threshold, and more pixels will be assigned to 255), but it leads to broken edges. A smaller window makes the algorithm more sensitive to local information, but the window size should not be too small. For an effective threshold calculation, the window should include both the foreground and background. A 7×7 window size and a constant of 1 are selected when applying the adaptive thresholding algorithm. This window size is selected so that it always include some foreground crystal and background picture. The constant 1 is chosen to avoid broken edges, as it affects the algorithm for separating crystals. With this constant, the majority of details are preserved, but the majority of noise is also preserved. Fortunately, the noise is not present within the crystals themselves. There is always some white space surrounding each cluster of crystals that does not contain any noisy pixels. We can effectively remove the noise carried by the adaptive thresholding algorithm using the mask which is described in the next section.

$$dst(x,y) = \begin{cases} maxValue & src(x,y) > T(x,y) \\ 0 & otherwise \end{cases} \quad (2)$$

Mask Generation

The purpose of using the mask is to determine the location of all the crystal clusters and to filter out all the regions that do not belong to crystal clusters, thus effectively removing the noise (since they are not crystals). In the first step of the mask generation, Canny edge detection is used to locate all of the crystal edges. Edge detection detects edges based on changes in intensity, which generates an edge map containing thin lines and no

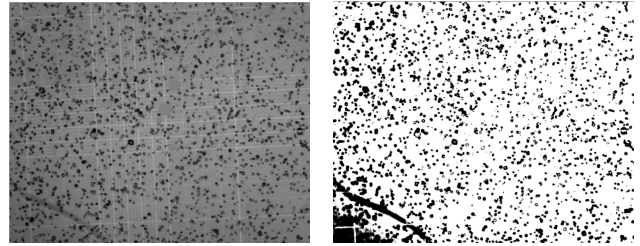


Figure 5: (a) Image after the bubbles are detected and removed (b) Corresponding Otsu's thresholding results applied on the image, with the lower left corner incorrectly segmented.

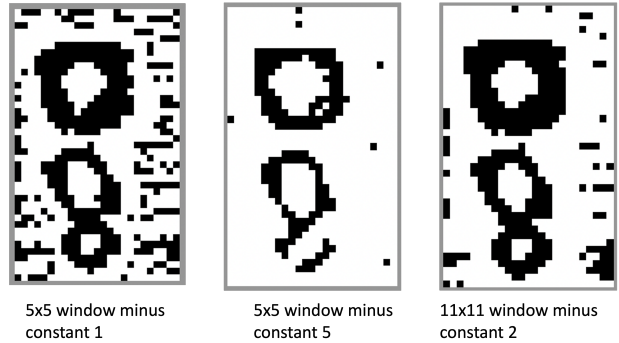


Figure 6: Different choices of window size and constant affect the results of adaptive Gaussian thresholding.

areas. To close small gaps in the edge map, we use morphological closing, which involves performing a dilation operation followed by an erosion operation. The window size we choose for closing is 3×3 , since the edges are thin. Next, we use flood filling to create areas in the image. We observe that the closing graph may contain some additional lines that stem from auxiliary lines in the original microscopic image used by manufacturers to aid manual counting. These lines overlap with the crystals and may affect our crystal separation results. To remove the unwanted lines, we use morphological opening, which involves an erosion operation

followed by dilation.

After performing all of these steps, we obtain a mask that identifies the locations of the crystal clusters. This mask can then be applied to the image after adaptive thresholding to eliminate noise present in that image. In order to improve the accuracy of the crystal separation and counting algorithm, we take the additional step of removing any small dots on the image that have an area of five pixels or less using connected components (connectivity = 8). These dots originate from the original microscopic image rather than from the adaptive thresholding algorithm, and are typically too small and lack openings to be considered as crystals. Following this step, we obtain a binary image where the crystal clusters are clearly visible in the foreground, and the noise is greatly reduced. The next step in the process is to perform the crystal separation algorithm.

Four-pointer Search

Our crystal separation algorithm is named “four-pointer search” as it employs four pointers to track the centers of the crystals and perform the necessary cuts to separate them. This search algorithm is based on the assumption that all crystals have an opening at their center. We use the cluster of crystals in Figure 7 as an example to demonstrate how the crystal separation process works. Firstly, we locate the centroids of all crystals using connected components. Then, we initiate four pointers at the topmost, leftmost, rightmost, and bottommost centroids. It is possible for two or more pointers to point at the same centroid, but this is not an issue, we always begin with four pointers.

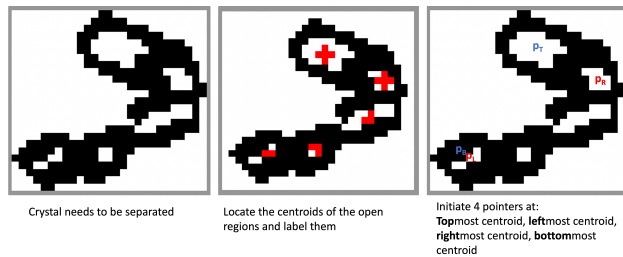


Figure 7: Initiating four-pointer search.

The four pointers can be grouped into two pairs: the left and right pointers form one pair, and the top and bottom pointers form the other. In order to avoid clutter, I will only describe the movement of the left and right pointers in detail, but note that the top and bottom pointers move in a similar fashion.

As shown in Figure 8, the pointers start at the edges and move towards the center. On the x -axis, the left pointer moves right towards the nearest centroid while the right pointer moves left towards the nearest centroid. Similarly, the top and bottom pointers move down and up, respectively, towards the nearest centroids on the y -axis. The pointers continue moving until they meet each other.

Figure 9 (a) shows the path taken by the left and right pointers in red, and the path taken by the top and bottom pointers in blue. The centroids where the top and bottom pointers meet, and where the left and right pointers meet, respectively, are highlighted in yellow circles in Figure 9 (b). Instead of making cuts immediately, the four-pointer search algorithm stores information during the search process and makes cuts after all the searches are complete. In Figure 9 (c), the necessary cuts are shown as red and

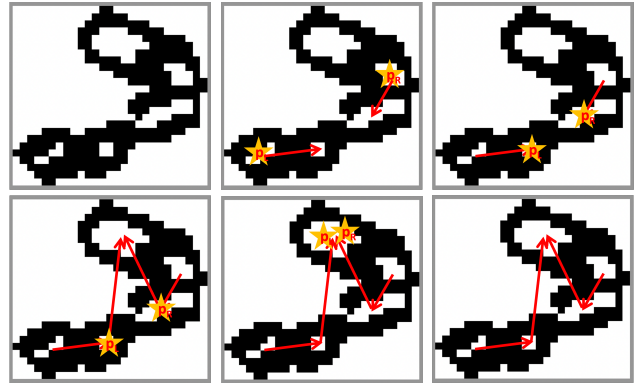


Figure 8: Searching process of the left and right pointers.

blue dotted lines. These cuts are generated by moving the pointers during the search process. If a new cut is required, the algorithm records three pieces of information: the slope and midpoint of the cut, and which side the centroid is on.

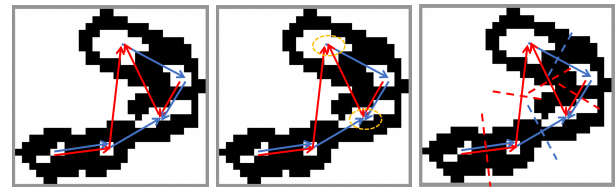


Figure 9: (a) Plots of the searching path for the left and right pointers in red, and for the top and bottom pointers in blue. (b) The two centroids where the left and right and top and bottom pointers meet, respectively, are highlighted in yellow. (c) The cuts that need to be made are demonstrated.

To further elaborate on the process of recording information during the search process, we present an example in Figure 10. Consider the cut shown in the blue color in Figure 10 (a), involving the two centroids highlighted in Figure 10 (b). To record the information, we create a dictionary for each of the centroids. Firstly, the slope of the cut is recorded, as shown in Figure 3(d). Secondly, the mid-point of the cut is recorded, as illustrated in Figure 3(e). Finally, the side of the centroid (i.e., whether it belongs to the top, left, bottom or right side) is recorded in a machine-readable format, with the notation 0, 1, 2, and 3 representing top, left, bottom, and right, respectively. This information is then used to generate the necessary cuts after the search process has been completed.

Once the search process is completed, all the necessary information for separating each crystal has been gathered. To perform the separation, we apply the cuts for each centroid based on the slope, mid point, and direction information that was recorded during the search. For instance, to separate the crystal highlighted in the yellow dot in Figure 11 (a), we must consider the three cuts that are related to it in Figure 11 (b). Using the midpoint, slope, and side information gathered earlier, we can make the appropriate cuts. To associate the all pixels in the image to the correct crystal, the algorithm checks each pixel in the vicinity of the crystal cluster and assigns it to the corresponding crystal based on the location of the pixel with respect to the cuts. In Figure 11 (c), the area above the first cut is considered as the candidate region for the current crystal. In Figure 11 (d), the second cut further nar-

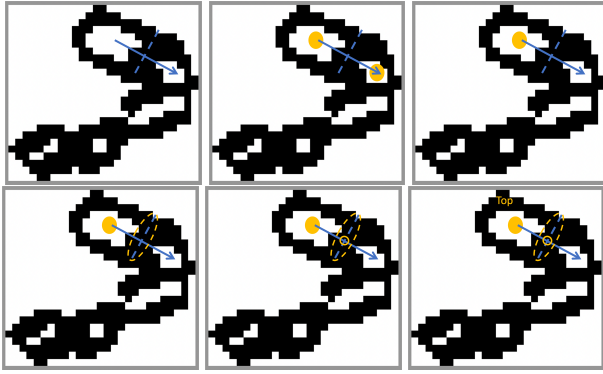


Figure 10: (a) The sample blue colored cut. (b) Highlight the two centroids related to this cut with yellow dots. (c) Use the top centroid as an example. (d) Record the slope. (e) Record the mid point. (f) Record the side (top).

rows down the qualified region to only the pixels on the left side of the cut. Finally, the third cut in Figure 11 (e) limits the qualified region to only the pixels on the right side of the cut. These three cuts ensure that only the pixels within the area shown in Figure 11 (f) is selected for the sample crystal.

The cuts made by left/right pointer only assign pixels to left/right side, and the cuts made by top/bottom pointer only assign points to the top/bottom centroids. More specifically, how the algorithm assign a pixel to the correct side is as follows:

If $x < midx + \frac{y-midy}{slope}$, the pixel belongs to the left centroid, otherwise it belongs to the right centroid.

If $y < midy - \frac{x-midx}{slope}$, the pixel belongs to the top centroid, otherwise it belongs to the bottom centroid.

This cutting process is repeated for each crystal, with the algorithm checking each pixel and assigning it to the corresponding crystal based on the cut information.

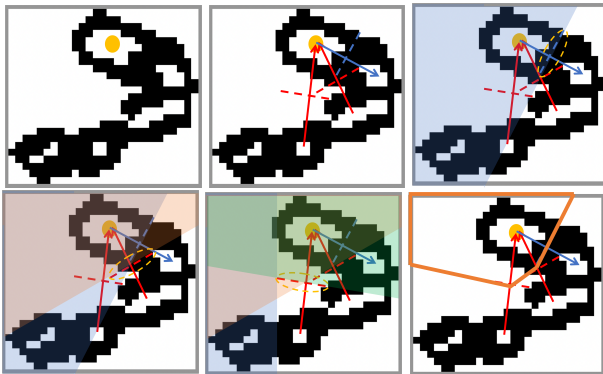


Figure 11: The process of separating the crystal highlighted using the yellow dot.

Once all pixels have been assigned, each crystal can be represented using a separate color, with the pixels belonging to that crystal being set to that specific color and the pixels belonging to background set to black, as shown in Figure 12. The success cases are shown on the top of Figure 12, where the crystals are separated accurately using the four-pointer search algorithm. On the bottom of Figure 12, some failure cases are shown. In the

failure example on the left, the crystal in the middle has a broken edge, and therefore the algorithm mistakenly identifies four crystals instead of five. In the failure example on the right, there is no opening center on the top crystal, causing the algorithm to regard the two crystals on the top as a single crystal. These failure cases are stem from the fact the current algorithm assumes that all the crystals have an opening center. In the future, efforts will be made to enhance the robustness of the algorithm and handle these boundary cases.

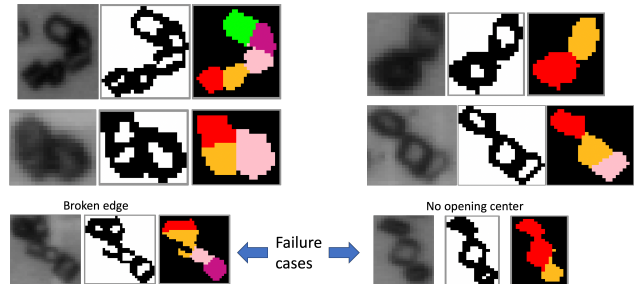


Figure 12: Color coded crystal separation results, showing both success and failure cases.

User Interface

Since the targeted users of the algorithm are manufacturers in the food industry, a user interface is developed. The interface allows users to simply select an image and receive the output of the segmented crystals without needing to understand the technical details of the algorithm. The running logic of the interface is based on the algorithm described earlier in this paper. This interface reduces the barrier of entry for non-technical users to utilize the algorithm, increasing its accessibility and potential impact in the industry. As shown in Figure 13, the user interface includes a main page where users can select crystal images, and adjust the parameters for the algorithm using the sliders if necessary. Then the user can proceed with the bubble detection and removal process by clicking the “Detect” button following the “Hide bubbles” button. The user can adjust the parameters if he or she is not satisfied with the bubble detection result. After the bubbles are removed, the user can trigger the crystal analysis algorithm by clicking the “Show results” button.

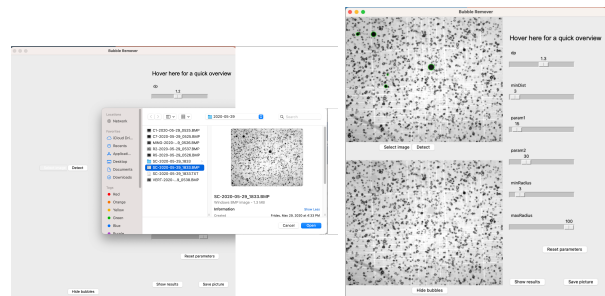


Figure 13: The way users interact with the GUI.

The algorithm will then run on the selected image and generate the separated crystals and analysis results as shown in Figure 14. The algorithm provides several useful analysis results for the manufacturers, including a histogram of crystal count versus crystal size, average number of crystals per window (using a window

size of 300×300 pixels as is used in manual counting), average crystal size, and percentage of the image area covered by crystals. Additionally, a graph with annotated crystal locations and numbers detected by the algorithm is presented for the manufacturer to verify the accuracy of the algorithm.

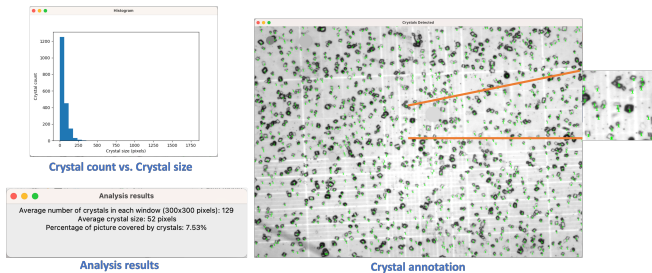


Figure 14: Analysis results generated by the GUI.

Future Work

As previously stated, this paper focuses solely on the seed count stage of crystal fabrication, during which the crystal seeds are relatively small. As the crystals grow larger in future stages, they may overlap, making it necessary to employ a more general approach such as deep learning to separate them. Our current plan is to use Mask R-CNN for this purpose, as it is capable of handling overlapped crystals. However, as we do not have any ground truth masks from the manufacturer, we must generate the masks required to train the Mask R-CNN model. We have already trained a CycleGAN model to generate synthetic masks. This idea of using CycleGAN to generate masks is also mentioned in [21]. Future work will be devoted to completing the entire model. The logic flow of the deep learning approach is shown in Figure 15.

In summary, this paper introduces an entire pipeline that can automatically detect and remove air bubbles from microscopic images of crystals, and obtain information on crystal count, crystal size and other relevant information, thereby saving a considerable amount of time and effort involved in manual inspections. The algorithm includes pre-processing steps, wherein the Hough circle method is used to detect air bubbles, then a mask is applied and adaptive thresholding is utilized to obtain a clean crystal binary image. Subsequently, the “four-pointer search” algorithm is employed to separate all the individual crystals. Additionally, the paper suggests that for future stages where crystals may overlap, a more general approach like mask R-CNN can be used. Overall, the proposed image analysis framework can greatly improve the efficiency and accuracy of crystal analysis in the food industry.

Acknowledgement

This manuscript is based upon work supported by the SMART consortium, Purdue University, and by the National Science Foundation under Award No. 2134667. Any opinions, findings, conclusion, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the view of the SMART consortium or the National Science Foundation.

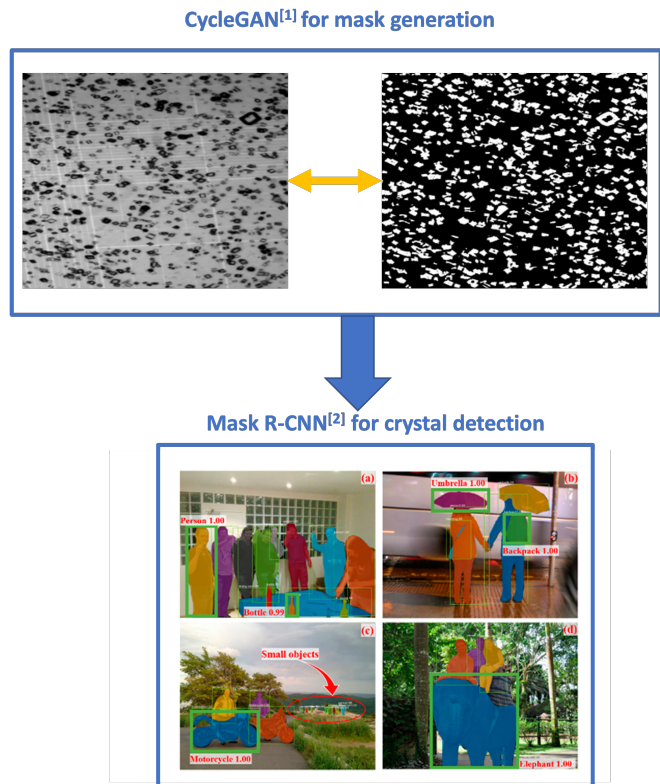


Figure 15: Deep learning approach logic flow.

References

- [1] T. Halawani, R. Mohler, and W. Kolodziej, “A two-step bilinear filtering approximation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 344–352, 1984.
- [2] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.
- [3] E. R. Davies, *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.
- [4] T. R. Singh, S. Roy, O. I. Singh, T. Sinam, K. Singh, *et al.*, “A new local adaptive thresholding technique in binarization,” *arXiv preprint arXiv:1201.5227*, 2012.
- [5] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.
- [6] R. M. Haralick, S. R. Sternberg, and X. Zhuang, “Image analysis using mathematical morphology,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532–550, 1987.
- [7] C. Li, H. Chen, X. Li, N. Xu, Z. Hu, D. Xue, S. Qi, H. Ma, L. Zhang, and H. Sun, “A review for cervical histopathology image analysis using machine vision approaches,” *Artificial Intelligence Review*, vol. 53, pp. 4821–4862, 2020.
- [8] J. Angel Arul Jothi and V. Mary Anita Rajam, “A survey on automated cancer diagnosis from histopathology images,” *Artificial Intelligence Review*, vol. 48, pp. 31–81, 2017.
- [9] B. Chaudhuri, K. Rodenacker, and G. Burger, “Characterization and featuring of histological section images,” *Pattern*

- Recognition Letters*, vol. 7, no. 4, pp. 245–252, 1988.
- [10] J. Sudbø, R. Marcelpoil, and A. Reith, “Caveats: numerical requirements in graph theory based quantitation of tissue architecture,” *Analytical Cellular Pathology*, vol. 21, no. 2, pp. 59–69, 2000.
 - [11] B. Weyn, W. Tjalma, P. Vermeulen, A. Van Daele, E. Van Marck, and W. Jacob, “Determination of tumour prognosis based on angiogenesis-related vascular patterns measured by fractal and syntactic structure analysis,” *Clinical Oncology*, vol. 16, no. 4, pp. 307–316, 2004.
 - [12] C. MacAulay and B. Palcic, “An edge relocation segmentation algorithm,” *Analytical and quantitative cytology and histology*, vol. 12, no. 3, pp. 165–171, 1990.
 - [13] M. Guillaud, D. Cox, K. Adler-Storthz, A. Malpica, G. Staerckel, J. Matisic, D. Van Niekerk, N. Poulin, M. Follen, and C. MacAulay, “Exploratory analysis of quantitative histopathology of cervical intraepithelial neoplasia: Objectivity, reproducibility, malignancy-associated changes, and human papillomavirus,” *Cytometry Part A: The Journal of the International Society for Analytical Cytology*, vol. 60, no. 1, pp. 81–89, 2004.
 - [14] P. Guo, H. Almubarak, K. Banerjee, R. J. Stanley, R. Long, S. Antani, G. Thoma, R. Zuna, S. R. Frazier, R. H. Moss, *et al.*, “Enhancements in localized classification for uterine cervical cancer digital histology image assessment,” *Journal of pathology informatics*, vol. 7, no. 1, p. 51, 2016.
 - [15] V. Torre and T. A. Poggio, “On edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 147–163, 1986.
 - [16] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, “Design of an image edge detection filter using the Sobel operator,” *IEEE Journal of solid-state circuits*, vol. 23, no. 2, pp. 358–367, 1988.
 - [17] Y. Wang, D. Crookes, O. S. Eldin, S. Wang, P. Hamilton, and J. Diamond, “Assisted diagnosis of cervical intraepithelial neoplasia (CIN),” *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 1, pp. 112–121, 2009.
 - [18] C. Wählby, I.-M. Sintorn, F. Erlandsson, G. Borgefors, and E. Bengtsson, “Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections,” *Journal of microscopy*, vol. 215, no. 1, pp. 67–76, 2004.
 - [19] A. Bieniek and A. Moga, “An efficient watershed algorithm based on connected components,” *Pattern recognition*, vol. 33, no. 6, pp. 907–916, 2000.
 - [20] D. M. Montserrat, Q. Lin, J. Allebach, and E. J. Delp, “Training object detection and recognition CNN models using data augmentation,” *Electronic Imaging*, vol. 2017, no. 10, pp. 27–36, 2017.
 - [21] L. Wu, A. Chen, P. Salama, K. W. Dunn, and E. J. Delp, “An ensemble learning and slice fusion strategy for three-dimensional nuclei instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1884–1894, 2022.
 - [22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.
 - [23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-

CNN,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

- [24] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

Author Biography

Qiyue Liang received her BS in electrical engineering from Purdue University (2018). She is currently working towards her PhD in electrical engineering at Purdue. Her research is focusing on image algorithms, materials science and biomedical image analysis, machine learning and deep learning.