Can Tho University Journal of Science
website: ctujs.ctu.edu.vn

DOI: 10.22144/ctu.jen.2022.006

Build coconut counting system using image technology

Nguyen Huu Quang[1], Truong Quoc Bao[2*] and Ngo Quang Hieu[2]

[1]*Engineer of Automation, Course 22, College of Engineering Technology, Can Tho University, Viet Nam*

[2]*College of Engineering Technology, Can Tho University, Viet Nam*

[*]*Correspondence: Truong Quoc Bao (email: tqbao@ctu.edu.vn)*

Article info.

Received 28 May 2021
Revised 10 Jun 2021
Accepted 24 Jul 2021

Keywords

Count coconuts, conveyor, distance transform, morphological operations, watershed segmentation

ABSTRACT

In our country today, the counting of dried coconuts at the production facilities is done manually, takes a lot of time and is not accurate. The goal of this study is to build an automatic, fast and accurate coconut counting system. The study was conducted on the peeled dried coconut fruit with a diameter of 15 cm to 20 cm using image processing technology and open-source computer vision library - OpenCV library. The algorithm includes four main steps. First, determine the object and the background using the Otsu segmentation method. Next, estimate the distance between the background and the object to determine the closest area to the center of the object. Then, find the contour, determine the center and area of the object to reduce the noise. The watershed segmentation algorithm is used to separate overlapping and stacking objects. Finally, count the number of objects contained in the image. In the initial experimental results, the counting system has had an accuracy of over 95% with processing time per image about 75 ms and the counting capacity of the system is over 2000 fruits/hour has confirmed the efficiency of the proposed method.

1. INTRODUCTION

Coconut, a popular tree especially grown in the Mekong River Delta and the Central Coast currently has about 150,000 hectares of coconut land with annual coconut production of over one billion fruits, is mainly distributed in Ben Tre, Tra Vinh, Kien Giang and Ca Mau. In which, Ben Tre and Tra Vinh thrive in both cultivated areas and processing export. The annual total export of coconut products of Viet Nam is about over 200 million USD. However, coconut is often grown and processed on a small scale, less investment in machinery and equipment leads to low productivity (Vu Trung, 2014).

Currently, the work of counting dried coconuts at the purchasing and processing establishment is done manually, easy to lose and difficult for employees when they have to count and memorize for a long time. In addition, manual counting also increases production costs for businesses. It requires a lot of labor if they want the fruit counting to take place quickly. If the fruit import is slow, it will affect the processing later. As a result, the automation of the coconut fruit counting process is an essential need. However, at present, there are not many publications on research, the application of technologies in estimating and counting coconut fruit. Studies often stop at counting the number of fruits on the tree image (Wijethunga et al., 2008; Malik et al., 2016; Chi Cuong et al., 2017) or estimate crop yield (Dorj et al., 2013; Behera et al., 2019). Some experimental systems have relatively high investment costs and are only suitable for large,

clearly planned farming areas. Whilst in Viet Nam there mainly is small-scale agriculture and no planning, it is difficult to apply (Häni et al., 2020). In our country, studies applying these applications are still not popular and primarily use manual counting of workers, so this study is very essential and practical.

In this paper, a new image processing algorithm is proposed to count the number of dried coconuts. First, the image obtained from the system is preprocessed and uses the Otsu segmentation method to separate the object and the background. Next, the area near the center of the object is determined using the distance transform technique. Then, the algorithm find contour of all objects in the image. Finally, the watershed segmentation algorithm used to separate overlapping and touching objects. The general processing process of the method is presented in Figure 1.

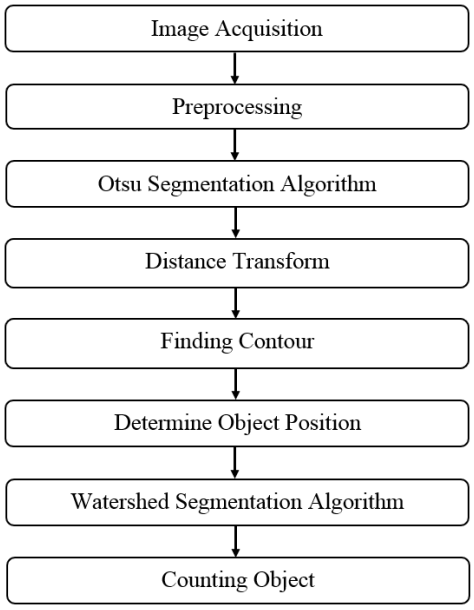


Figure 1. Overview block diagram of the proposed method

2. MATERIAL AND METHODS

2.1. Image acquisition and preprocessing

Electromechanical system includes two main components: the conveyor and the starter. The conveyor belt was designed as a rubber conveyor belt with a length of 2 m, width of 0.9 m, and height of 1 m. The velocity of the conveyor is 5 cm/s with counting productivity over 2000 fruits/hour, as shown in Figure 2. In this study, the conveyor was designed with the fixing velocity, unchanged. The starter holds the duty of closing, interrupting, and protecting motor overload. The main components of the starter includes: circuit breaker one phase two pole (CB), Contactor, thermal relay, fuse, wire, etc.

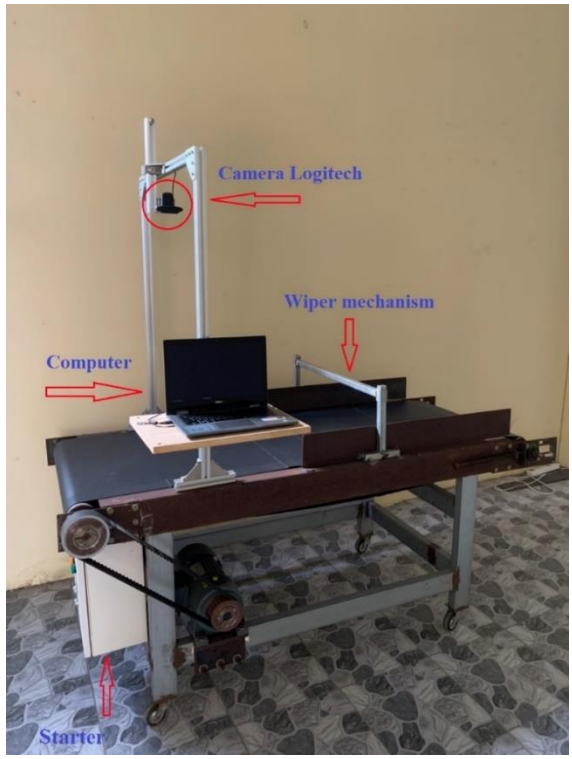


Figure 2. Coconut dried fruit counting model

The camera was connected directly to the computer via the USB 3.0 port. Dried coconut is continuously poured on the conveyor belt. Then it is moved into the capture area and image acquisition by Logitech HD Pro C922 camera mounted perpendicular to the conveyor surface. In this system, a shelf is designed to support the fixing of the camera and the distance from the camera to the conveyor surface is 0.9 m. Adjust the camera position so that the system captures the input image only the black background of the conveyor surface. This created the advantage condition for image processing, as shown in Figure 3. Next, the input image is used to reduce the noise using a Gaussian filter with mask size 5x5 and enhance the contrast with the adaptive histogram equalization method.

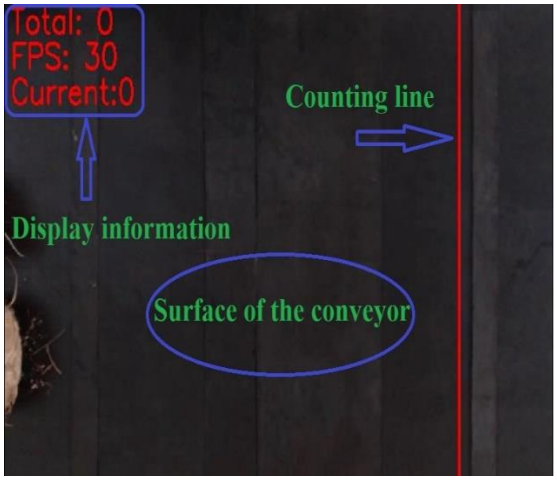


Figure 3. Input image of the system

2.2. Image Segmentation

Input images after the acquisition and preprocess are segmented to determine the object and the background using Otsu segmentation method. The Otsu algorithm (Otsu, 1979) was launched by Nobuyuki Otsu in 1979. The objective of the algorithm is a threshold T, which is calculated automatically based on grayscale values of the pixels in order to replace the use of fixed segment in the problem binary image based on threshold grayscale.

The basic content of the method is described as follows: Firstly, convert the input images to grayscale images and statistics on the number of gray level, assuming there is $L(0 \le L \le 255)$ grayscale in the image. Then, dichotomize the pixels into two classes C_1 and C_2 (background and objects, or vice versa) by a threshold at level T; C_1 denotes pixels with levels $[1..T]$, and C_2 denotes pixels with levels $[T+1..L]$. The total number of pixels is called $N, h[i]$ number of pixels in the gray level $i(0 \le i \le 255)$ and probability of appearance grayscale level i is: $p_i = \frac{h[i]}{N}$. From that, calculate the optimal threshold T^* by the formula (1).

$$T^* = Arg \underset{0 \le T < L}{Max} \left\{ \sigma_B^2(T) \right\} \quad (1)$$

$\sigma_B^2(T)$ is identified by formula from $(2) \sim (6)$.

Variance:

$$\sigma_B^2(T) = \omega_1(T)\omega_2(T)\left(\mu_2(T) - \mu_1(T)\right)^2 \quad (2)$$

Probability appears of C_1:

$$\omega_1(T) = P_1 = \sum_{i=0}^{T-1} p_i \quad (3)$$

Probability appears of C_2:

$$\omega_2(T) = P_2 = \sum_{i=T}^{L-1} p_i = 1 - P_1 \quad (4)$$

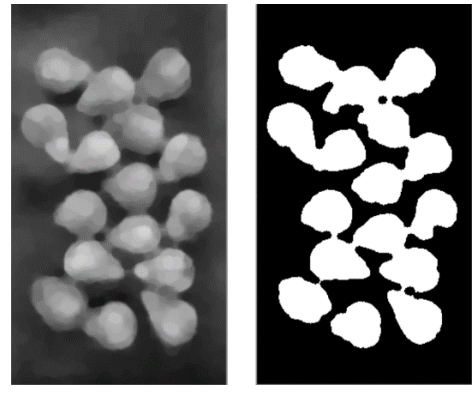
The average gray level of C_1:\

$$\mu_1(T) = \sum_{i=0}^{T-1} iP(i/C_1) = \sum_{i=0}^{T-1} ip_i / \omega_1(T) \quad (5)$$

The average gray level of C_2:

$$\mu_2(T) = \sum_{i=T}^{L-1} iP(i/C_2) = \sum_{i=T}^{L-1} ip_i / \omega_2(T) \quad (6)$$

The Otsu algorithm is one of the image segmentation method which is a simple algorithm to compute the threshold T in order to solve the global threshold, as shown in Figure 4. Therefore, T is the important factor that decides the success or failure of this algorithm.



(a) *(b)*

Figure 4. Segmentation results using Otsu algorithm

(a) Enhancement image and
(b) Otsu segmentation algorithm

2.3. Distance transform

The distance transform operator generally takes binary images as inputs. In this operation, the gray level intensities of the points inside the foreground regions are changed to distance their respective

distances from the closest 0 value, as shown in Figure 5.

The distance transform $D(i, j)$ of a binary image $b(i, j)$ is defined as follows. Let $d(k,l)$ be some *distance metric* between pixel offsets. Two commonly used metrics include the *city block* or *Manhattan* distance (Szeliski, 2010).

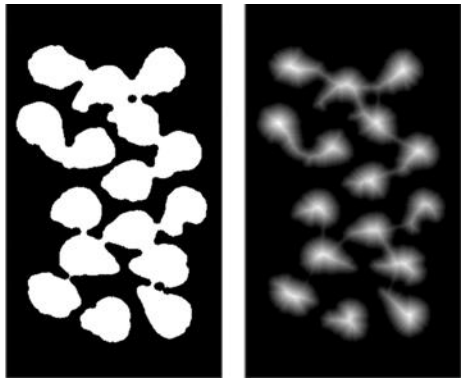
$$d_1(k,l) = |k| + |l| \quad (7)$$

And the Euclidean distance

$$d_2(k,l) = \sqrt{k^2 + l^2} \quad (8)$$

The distance transform is defined as

$$D(i, j) = \min_{k,l:b(k,l)=0} d(i-k, j-l) \quad (9)$$



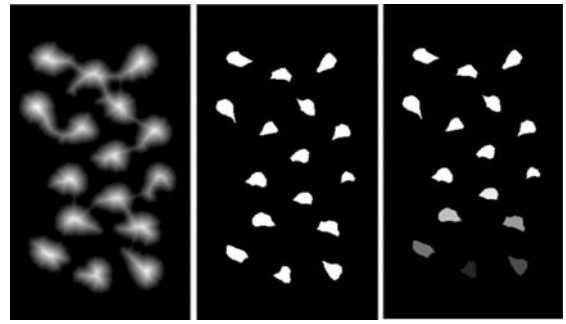
(a) *(b)*

Figure 5. Distance transform results

(a) Binary image and (b) distance transform image

2.4. Create marker for watershed algorithm

After the distance transform image between the object and the background is found, the binary image using the Otsu segmentation method is performed to determine the nearest center area of each object. Then, the findContours() function in the OpenCV library is performed to find the contour and calculate the total number of objects in the image, as shown in Figure 6.



(a) *(b)* *(c)*

Figure 6. Create marker image for watershed algorithm

(a) Distance transform image, (b) binary image, and (c) marker image

2.5. Determine object position

After separating the object from the background and delimiting the object by the contours, we find the center of the object. To determine the coordinates of the object, we have to find the moment of the image that is shown in theoretical (Belkasim et al., 1991).

The mathematical equation of moment is shown in (10):

$$\mu_n = \int_{-\infty}^{+\infty} (x-c)^n f(x)\,dx \quad (10)$$

Where n^{th} is the moment around point c. When you apply on the 2D space, we have two independent variables to present (10) . That is presented again in (11) :

$$\mu_{m,n} = \iint (x-c_x)^m (x-c_y)^n f(x,y)\,dxdy \quad (11)$$

Where, $f(x, y)$ is the continuous function. So, we have to digitize each pixel that is shown in (12):

$$\mu_{m,n} = \sum_{x=0}^{\infty} \sum_{y=0}^{\infty} (x-c_x)^m (y-c_y)^n f(x,y) \quad (12)$$

After we calculate the region of the binary image, we have to calculate the 0^{th} moment.

$$\mu_{0,0} = \sum_{x=0}^{w} \sum_{y=0}^{h} x^0 y^0 f(x,y) \quad (13)$$

The formula is written again after x_0 and y_0 that is shown in (14).

$$\mu_{0,0} = \sum_{x=0}^{w} \sum_{y=0}^{h} f(x,y) \quad (14)$$

To determine the center of the object, we have to compute on two axes.

$$centroid = \left(\frac{\mu_{1,0}}{\mu_{0,0}}, \frac{\mu_{0,1}}{\mu_{0,0}} \right) \quad (15)$$

The total number of pixels will be aggregated and expressed as follows:

$$\begin{aligned} sum_x &= \sum\sum x f(x,y) \\ sum_y &= \sum\sum y f(x,y) \end{aligned} \quad (16)$$

After that, we get the average by dividing the total of number pixel. Its formula is shown in (17).

$$\begin{aligned} \mu_{1,0} &= \frac{sum_x}{\mu_{0,0}} \\ \mu_{0,1} &= \frac{sum_y}{\mu_{0,0}} \end{aligned} \quad (17)$$

For the function in OpenCV, the coordinate of the center is computed as follow:

$$\begin{aligned} c_x &= \text{int}\left(M\left["m10"\right] / M\left["m00"\right]\right) \\ c_y &= \text{int}\left(M\left["m01"\right] / M\left["m00"\right]\right) \end{aligned} \quad (18)$$

Determine the center of the object is shown in Figure 7.

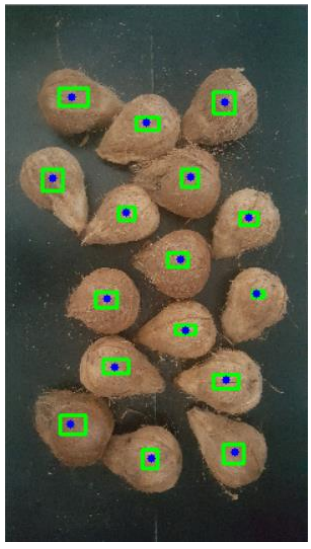


Figure 7. The resulting image determines the center of each object

2.6. Watershed segmentation algorithm

The algorithm introduced by Luc Vincent and Pierre Soille is based on the concept of "immersion". Each of local minima of a gray-scale image which can be regarded as a surface has a hole, and the surface is immersed out into water. Then, starting from the minima of the lowest intensity value, the water will progressively fill up different catchment basins of the image. Conceptually, the algorithm then builds a dam to avoid a situation that the water coming from two or more different local minima would be merged. At the end of this immersion process, each local minimum is totally enclosed by dams corresponding to watersheds of the image (Gonzalez et al., 2008).

Let $M_1, M_2, ..., M_r$ be sets denoting the coordinates of the points in the regional minima of an image $g(x,y)$. As indicated at the end of section (19), this typically will be a gradient image. Let $C(M_i)$ be a set denoting the coordinates of the points in the catchment basin associated with regional minimum M_i (recall that the points in any catchment basin form a connected component). The notation min and max will be used to denote the minimum and maximum values of $g(x,y)$. Finally, let $T[n]$ represent the set of coordinates (s,t) for which $g(s,t) < n$. That is,

$$T[n] = \{(s,t) \mid g(s,t) < n\} \quad (19)$$

Geometrically, $T[n]$ is the set of coordinates of points in $g(x,y)$ lying below the plane $g(x,y) = n$.

The topography will be flooded in integer flood increments, from $n = \min+1$ to $n = \max+1$. At any step n of the flooding process, the algorithm needs to know the number of points below the flood depth. Conceptually, suppose that the coordinates in $T[n]$ that are below the plane $g(x,y) = n$ are "marked" black, and all other coordinates are marked white. Then when we look "down" on the $xy-plane$ at any increment n of flooding, we will see a binary image in which black points correspond to points in the function that are below the plane $g(x,y) = n$. This interpretion is quite useful in helping clarify the following discussion.

Let $C_n(M_i)$ denote the set of coordinates of points in the catchment basin associated with minimum M_i

that is flooded at stage n. With reference to the discussion in the previous paragraph, $C_n(M_i)$ may be viewed as a binary image given by formula (20).

$$C_n(M_i) = C(M_i) \cap T[n] \quad (20)$$

In other words, $C_n(M_i) = 1$ at location (x, y) if $(x, y) \in C(M_i)\ AND\ (x, y) \in T[n]$; otherwise $C_n(M_i) = 0$.

Next, we let $C[n]$ denote the union of the flooded catchment basins at stage n:

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i) \quad (21)$$

Then $C[\max+1]$ is the union of all catchment basins:

$$C[\max+1] = \bigcup_{i=1}^{R} C(M_i) \quad (22)$$

It can be shown that the elements in both $C_n(M_i)$ and $T[n]$ are never replaced during the execution of the algorithm and that the number of elements in these two sets either increases or remains the same as n increases. Thus, it follows that $C[n-1]$ is a subset of $C[n]$. According to Eqs. (20) and (21), $C[n]$ is a subset of $T[n]$, so it follows that $C[n-1]$ is a subset of $T[n]$. From this we have the important result that each connected component of $C[n-1]$ is contained in exactly one connected component of $T[n]$.

The algorithm for finding the watershed lines is initialized with $C[\min+1] = T[\min+1]$. The algorithm then processes recursively, computing $C[n]$ from $C[n-1]$. A procedure for obtaining $C[n]$ from $C[n-1]$ is as follows. Let Q denote the set of the connected component in $T[n]$. Then, for each connected component $q \in Q[n]$, there are three possibilities:

1. $q \cap C[n-1]$ is empty.
2. $q \cap C[n-1]$ contains one connected component of $C[n-1]$.
3. $q \cap C[n-1]$ contains more than one connected component of $C[n-1]$.

Construction of $C[n]$ from $C[n-1]$ depends on which of these three conditions holds.

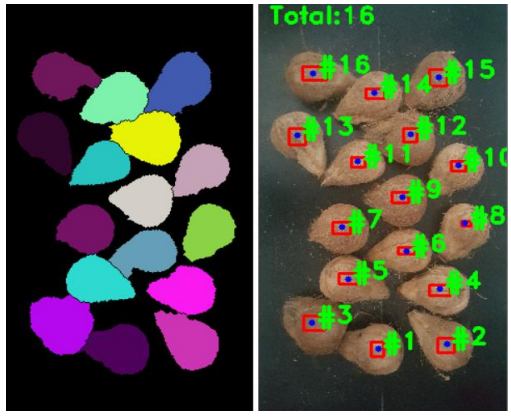
Condition 1 occurs when a new minimum is encountered, in which case connected component q is incorporated into $C[n-1]$ to form $C[n]$.

Condition 2 occurs when q lies within the catchment basin of some regional minimum, in which case q is incorporated into $C[n-1]$ to form $C[n]$.

Condition 3 occurs when all, or part, of a ridge separating two or more catchment basins, is encountered. Thus a dam (or dams if more than two catchment basins are involved) must be built within q to prevent overflow between the catchment basins. As explained in the previous section, a one-pixel-thick dam can be constructed when needed by dilating $q \cap C[n-1]$ with a 3×3 structuring element of 1s and constraining the dilation to q.

Algorithm efficiency is improved by using only values of n that correspond to existing intensity values in $g(x, y)$; we can determine these values, as well as the values of min and max, from the histogram of $g(x, y)$.

The watershed segmentation results is shown in Figure 8.



(a) *(b)*

Figure 8. Dried coconut counting results

(a) The watershed segmentation result and
(b) the counting result

3. RESULTS AND DISCUSSION

The system was experimented by pouring a certain amount of dried coconut on the conveyor. The dried coconut entered into image acquisition and processing region through the conveyor. The software program will display information on the monitor including counting line, total number of fruits counted, amount of fruits in each frame, and estimated center position of each fruit. when the dried coconut moves on the conveyor, their center position also moves until they touch the counting line, the total number of fruits will increase by one. However, the continuously pouring like that will cause overlap between of many coconuts, path to incorrect results, the data sheet is presented in Table 1. Therefore, the study has designed a wiper mechanism to solve this problem. The wiper mechanism helps to spread the dried coconut evenly on the conveyor surface. The experimental results show that the accuracy of the system is greatly increased when it is supported by the wiper mechanism.

Table 1. The experimental results of the system doesn't have wiper mechanism

The actual number (fruit)	The counting system (fruit)			Average error rate (%)	Average accuracy (%)
	1st	2nd	3rd		
4	1	1	2	66	34
9	4	4	5	51.85	48.15
12	5	4	5	61.1	38.9

To evaluate the effectiveness of the proposed method, the system was experimented with 24 count times. The number of fruits needs to be counted from 10 to 17 and done with three different count times, then compared with the actual count. The summary data sheet is presented in Table 2.

Table 2. The experimental results of the system have wiper mechanism

The actual number (fruit)	The counting system (fruit)			Average error rate (%)	Average accuracy (%)
	1st	2nd	3rd		
10	10	10	10	0	100
11	11	11	11	0	100
12	12	12	12	0	100
13	13	13	12	2.56	97.44
14	13	14	14	2.36	97.64
15	15	15	14	2.2	97.8
16	16	15	15	4.16	95.84
17	16	17	16	3.93	96.07

From the statistical results in Table 1 and Table 2, the study found that when the system does not support the wiper mechanism, the counting results have low accuracy of less than 50% and the lowest 34%. But when the system is supported by the wiper mechanism, the accuracy is significantly improved with an accuracy of over 95% and the highest up to 100%. Therefore, for the system to operate with high accuracy, it is necessary to have a wiper mechanism support.

The experiment is conducted with the rubber conveyor model and the velocity of conveyor belt is 5 cm/s. The photos of dried coconut are captured with Logitech Full HD Pro C922 camera in normal daylight conditions. The videos are recorded in AVI format with a resolution of 640x480. The program runs on Dell Inspiron 5379 computer equipped with Intel® Core ™ i7-8550U CPU processor, 16GB RAM. The counting program runs on Windows 10 Pro operating system with Visual Studio 2019 programming environment, Visual C ++ programming language, and open-source computer vision library OpenCV 3.4.9. The frame rate of the system is 12 frames per second The experimental results of the system are presented in Figure 9.

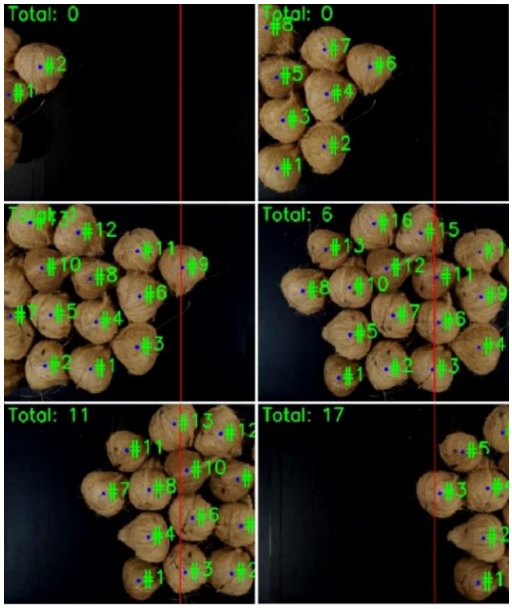


Figure 9. Counting results of the system when dried coconut moves on the conveyor from left to right

4. CONCLUSIONS

This study has proposed an efficient processing algorithm to automatically count the number of dried coconuts moving on the conveyor belt. The results of this study can be a prerequisite for the development of an automated agricultural product counting and classification system. The experimental results show that the average accuracy of the proposed method is over 95% with the productivity count over 2,000 fruits/hour. Therefore, it can be applied to the problem of counting the dried fruit coconuts moving on the conveyor belt.

In the future, further studies should be in upgrading hardware in combination with the research and the development of software algorithms to reduce processing time as well as increase the accuracy and productivity of the system. At the same time, design and experiment the automatic agricultural product counting and classification system.

REFERENCES

Behera, S. K., Pattnaik, A., Rath, A. K., Barpanda, N. K., & Sethy, P. K. (2019). Yield estimation of pomegranate using image processing techniques. *Int J Innov Technol Explor Eng, 8*(6S), 798-803.

Belkasim, S. O., Shridhar, M., & Ahmadi, M. (1991). Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognition*, *24*(12), 1117-1138.

Dorj, U. O., Lee, M., & Han, S. (2013). A Counting Algorithm for Tangerine Yield Estimation. *Center for Advanced Image and Information Technology, School of Electronics & Information Engineering, Chon Buk National University*.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (3rd ed.). Pearson Prentice Hall.

Häni, N., Roy, P., & Isler, V. (2020). A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, *37*(2), 263-282.

Malik, Z., Ziauddin, S., Shahid, A. R., & Safi, A. (2016). Detection and counting of on-tree citrus fruit for crop yield estimation. *IJACSA International Journal of Advanced Computer Science and Applications*, *7*(5), 519-523.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62-66.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Tran, C. C., Nguyen, D. T., Le, H. D., Truong, Q. B., & Truong, Q. D. (2017). Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis. In *2017 4th NAFOSTED Conference on Information and Computer Science* (pp. 132-137). IEEE.

Vu Trung. (2014). Development of the Coconut Industry. *STINFO Journal for Science and Technology Information*, 10, 4-10.

Wijethunga, P., Samarasinghe, S., Kulasiri, D., & Woodhead, I. (2008). Digital image analysis based automated kiwifruit counting technique. In *2008 23rd International Conference Image and Vision Computing New Zealand* (pp. 1-6). IEEE.