Team Number:	apmcm2305635
Problem Chosen:	A

2023 APMCM summary sheet

In response to labor challenges in China's apple industry, the world's largest producer, this paper presents a novel apple image recognition model. Designed to automate and estimate fruit counts before harvesting can enhance the sorting and picking processes and provide useful information for logistics planning. this model efficiently distinguishes apples from similar fruits, advancing technological innovation in agriculture and reducing the need for manual labor, potentially revolutionizing China's agricultural sector.

Aiming at Problem 1, we have designed the Tri-Harmonize Augmentation and Contours Detection to identify the count of apples. This method ensures precise apple counting by generating circles around each identified fruit, leading to an approximate of 28,073 apples detected throughout the dataset.

Aiming at Problem 2, image segmentation techniques will isolate the apples from the background, and object localization methodologies will be applied to pinpoint their exact positions, culminating in a 2D scatter diagram to visually map their spatial distribution.

Aiming at Problem 3, bicubic interpolation for image scaling and GrabCut for precise background removal are being utilized. Feature extraction is conducted via KMeans clustering, analyzing RGB values within central circular segments. The classification of apple maturity is then visualized in a histogram, effectively demonstrating the method's capability in discerning ripeness levels from image data.

Aiming at Problem 4, we manipulate the RGB values of pixels in each image to calculate the two-dimensional area of the apples. This area is then adjusted according to the image's scaling ratio and multiplied by a density factor of 0.7 to estimate mass.

Aiming at Problem 5, In solving Problem 5, a Convolutional Neural Network (CNN) is trained using a designated dataset for the task of apple recognition. This model is then applied to a separate set of images for identifying apples. Following this, a histogram is generated to visually depict the frequency of 11,241 identified apples, effectively illustrating their distribution within the dataset.

Keywords: Object Recognition Image Segmentation Image Classification Object Localization CNN

Contents

1. Introduction **1**
1.1 Background 1
1.2 Restatement of the Problem 1
2. Analysis of the Problem **2**
2.1 Analysis of the Problem 1 2
2.2 Analysis of the Problem 2 2
2.3 Analysis of the Problem 3 3
2.4 Analysis of the Problem 4 3
2.5 Analysis of the Problem 5 3
3. Models **4**
3.1 Model Establishment and Solution of Problem 1 4
3.1.1 *Tri-Harmonize Augmentation* 4
3.1.2 *Setup and Color Range Definition* 4
3.1.3 *Image Processing Loop* 4
3.2 Model Establishment and Solution of Problem 2 6
3.2.1 *Image Filtering using Gaussian Filtering* 6
3.2.2 *Salt and Pepper Noise* 7
3.3 Model Establishment and Solution of Problem 3 9
3.3.1 *Image Segmentation using GrabCut* 9
3.3.2 *RGB Color Space Model* 9
3.3.3 *Iterative Energy Minimization Segmentation Algorithm* 10
3.4 Model Establishment and Solution of Problem 4 13
3.4.1 *Quantifying Apple Size Through Area Calculations* 13
3.5 Model Establishment and Solution of Problem 5 15
3.5.1 *The Recognition of Apples and Fruit Classification* 15
3.5.2 *ConvNet* 16
3.5.3 *Hyperparameters and Configurations* 17
3.5.4 *Training Loop* 17
4. Model Evaluation and Improvement **18**
4.1 Model Evaluation of Problem 1 18
4.2 Model Evaluation of Problem 5 18
4.2.1 *Strength and Weakness* 20

5. References **21**
6. Appendix **22**

I. Introduction

1.1 Background

China, the global leader in apple production and export, produces approximately 35 million tons annually. The Belt and Road Initiative (BRI) has expanded the export market for Chinese apples, with key destinations along the route including Vietnam, Bangladesh, the Philippines, and Indonesia. To address labor shortages during peak apple-picking seasons, China initiated research on apple-picking robots in 2011. Despite significant progress, deploying these robots in real-world orchard environments poses challenges, specifically in the accurate identification of obstacles during harvesting. This issue directly impacts the quality of the fruit and operational efficiency. Additionally, post-harvest stages, including the recognition and classification of apples, present substantial difficulties due to similarities with other fruits.

1.2 Restatement of the Problem

In order to accurately identify obstacles, the objective is to create a high-performance apple image recognition model with superior recognition rates, fast processing speeds, and exceptional accuracy. This involves the analysis and extraction of features from labeled fruit images.

Question 1: Counting apples

Utilizing the provided dataset(Attachment 1) of harvest-ready apples, extract image features, establish a mathematical model, and accurately count the number of apples in each image. Subsequently, generate a histogram illustrating the distribution of all apples in the provided dataset.

Question 2: Estimating the positions of apples

Based on the image dataset(Attachment 1) of harvest-ready apples presented in the provided data, determine the positions of the apples in each image, utilizing the left bottom corner of the image as the coordinate origin. Subsequently, create a two-dimensional scatter diagram representing the geometric coordinates of all apples in the given dataset.

Question 3: Estimating the maturity state of apples

Leveraging the image dataset of harvest-ready apples as outlined in the provided data, formulate a mathematical model to compute the maturity of apples in each image. Subsequently, generate a histogram depicting the distribution of maturity levels across

all apples in Attachment 1.

Question 4: Estimating the masses of apples

Using the provided dataset (Attachment 1) of harvest-ready apples, calculate the two-dimensional area of each apple with the bottom left corner of the image as the coordinate origin. Estimate the masses of the apples and create a histogram illustrating the distribution of masses across all apples in Attachment 1.

Question 5: The recognition of apples

Extracting image features and developing a model to recognize apples from a dataset in Attachment 2, apply it to identify apples in Attachment 3, and create a histogram showing the distribution of how many and which images are apples with Id numbers in Attachment 3.

II. Analysis of the Problem

2.1 Analysis of the Problem 1

The problem addressed in this section is the augmentation of a dataset and the processing of images for apple detection in Convlutional Neural Network(CNN), particularly for image processing task. The primary objectives are to generate augmented images by blending different types of noise and sharpness, define color ranges in the HSV color space to filter apples based on their color and maturity state, and perform image processing to detect apples in both original and augmented images. This involves steps such as image conversion, creating masks for apple detection based on color ranges, finding contours of apples, counting the detected apples, and storing their attributes in a structured format.

2.2 Analysis of the Problem 2

Our focus is on estimating the spatial distribution of apples within a dataset from Appendix 1 of images. We start by extracting the coordinates of each apple identified by a Convolutional Neural Network (CNN), ensuring that each fruit's location is accurately recorded. These coordinates are then used to plot a scatter diagram, which will visually depict the distribution and position of the apples across the various images.

2.3 Analysis of the Problem 3

This complex process begins with the extraction of apple locations using a Convolutional Neural Network (CNN), followed by precise image cropping while carefully avoiding boundary exceedances. To accommodate images of insufficient size, we upscale using bicubic interpolation, renowned for its ability to preserve image quality. The GrabCut algorithm is then employed to isolate the fruit from its background, ensuring a focused analysis. Feature extraction via KMeans clustering paves the way for maturity assessment, where RGB values of strategically plotted circular paths on the apples provide the necessary data. These values are compared against a predefined maturity spectrum, allowing us to categorize the apples accordingly. The culmination of our endeavor is the creation of a histogram, a graphical representation that succinctly displays the distribution of apple maturity levels across the sample set, thus providing a clear visual summary of our findings.

2.4 Analysis of the Problem 4

In order to estimate masses of apples,we begin by cropping the images based on apple detection data from a CNN, while carefully managing edge boundaries to ensure accuracy. Smaller images are scaled up using bicubic interpolation to standardize the size for consistent processing. The background is then removed via the GrabCut algorithm, rendering it black for a clear delineation of the apples. Feature extraction follows, using KMeans clustering with K set to 5. We compute RGB values of circular pixel arrays centered on the apples, ignoring black pixels to focus on the fruit. We identify the closest cluster of RGB values to each apple, which helps in estimating the apple's area by counting the corresponding pixels. This pixel count is then adjusted for the image's scaling to accurately represent the apple's mass.

2.5 Analysis of the Problem 5

Convolutional Neural Network (CNN) is tailored for effective fruit classification in this problem. CNN features three convolutional layers with 3×3 filters, crucial for extracting diverse features from RGB images, complemented by max pooling layers that reduce spatial dimensions while preserving essential information. The network also integrates two fully connected layers, the first with 128 neurons, and the second corresponding to five fruit classes. This structure ensures efficient data processing,

from feature extraction to classification. Additionally, specific configurations like batch size, dataset size, and training/test split are meticulously chosen to optimize the model's performance for the large dataset of over 20,000 images, ensuring accurate and efficient image classification.

III. Models

3.1 Model Establishment and Solution of Problem 1

3.1.1 *Tri-Harmonize Augmentation*

We design to augment the dataset by creating new images that combine different types of noise and sharpness. This can be beneficial in training robust machine learning models, especially in image processing tasks. We blends three different versions of an image (unsharp masked, with Gaussian noise, and with salt-and-pepper noise) to create an the augmented images.

3.1.2 *Setup and Color Range Definition*

We define color ranges in the HSV (Hue, Saturation, Value) color space for different states of apples (ripe red, raw red, ripe yellow, raw yellow, ripe green, raw green). These color ranges are used to create masks that filter out apples based on their color and maturity state.

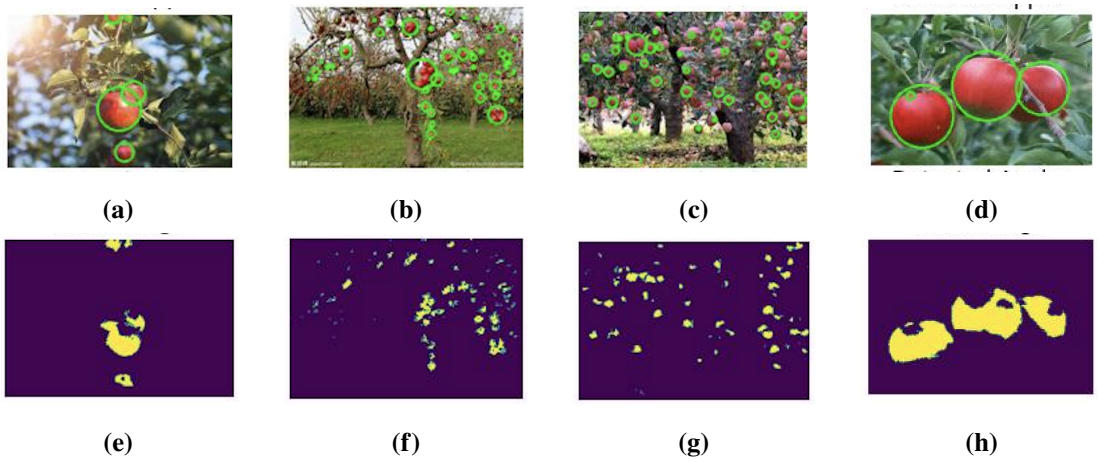
3.1.3 *Image Processing Loop*

For each image, we performs the following steps:

1. **Read and Convert Image:** The image is loaded in BGR (Blue, Green, Red) color space. Copies of the original image are made for later processing. The image is converted to both the RGB and HSV color spaces.
2. **Create Masks for Apple Detection:** For each predefined apple color range, a binary mask is created. This mask highlights areas of the image that fall within the specified color range. All individual masks are combined into a single mask that represents all apples regardless of their color or maturity.
3. **Find and Draw Contours:** The algorithm uses combined mask to detect contours, which are essentially boundaries of apples.For each contour, we find the smallest circle that can enclose the contour, which represents an apple.If the radius of this

circle is above a certain threshold (0.5 for original images, 1 for augmented images), it is counted as an apple, and the circle is drawn on the image.

4. **Count Apples:** The number of apples detected in each image is counted. For each apple detected in the augmented images, a dictionary with label, center point, and radius is created and added to a list.



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

(e) **(f)** **(g)** **(h)**

Fig 1. Count of Apples in Masked Image and Detected Image

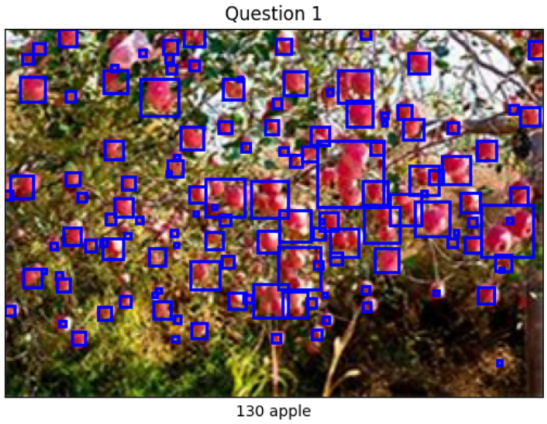


Fig 2. Numbers of Counted Apples

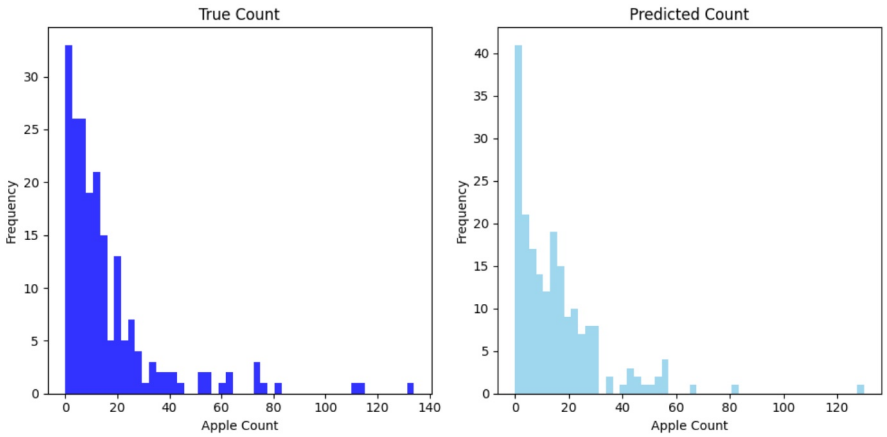


Fig 3. Histogram of Count Discrepancy in Apple Counting

The diagram compares two histograms that show the distribution of apple counts. The first histogram represents the actual number of apples counted, and it shows that lower apple counts are much more common than higher ones. The second histogram, labeled represents what a prediction model thinks the apple counts will be. This model's predictions are more spread out, indicating it expects there to be more instances of medium to high apple counts than there actually are. Therefore, while both histograms show that small apple counts are common, the prediction model does not capture the steep drop-off in higher apple counts seen in the actual data.

3.2 Model Establishment and Solution of Problem 2

3.2.1 *Image Filtering using Gaussian Filtering*

During identifying images, it is usually difficult for a model to capture and interpret a photo that may contain imperfections or variations. This is because while using deep learning model such as Convolutional Neural Networks(CNNs), can be overly sensitive to the exact features present in their training data. When the model encounters new images which contain slight variations, its performance can degrade because it has not learnt to recognize the underlying patterns amidst such noise.

As a solution, Gaussian noise will be added to the training images which stimulates the potential variations and imperfections that the model might encounter in this paper. Due to statistical properties of Gaussian noise, the model will be trained to become more generalized and be able to perform classification task and robust against minor perturbations.

Therefore, we add noise generated from a Gaussian distribution, defining the

characteristics of the noise: the mean μ and the standard deviation (σ). Then, $\mu$is set to zero in order to ensure that the noise does not systematically increase or decrease the pixel intensities of the images but rather adds variability around their original values. σ is chosen as 0.01 to 0.05 randomly in order to strike a balance between making the noise noticeable and not overwhelming the original image content. Later, noise is generated for individual images, creating a noise matrix I_i that matches the dimensions of the image N_i. Each element of N_i is drawn from the Gaussian distribution $N(0, \sigma 2)$, ensuring that the noise is both random and adheres to the defined statistical properties. Once the noise is generated, it is added to the original images to create augmented versions. This is done by simply adding the noise matrix N_i to each corresponding image I_i, resulting in a new image $I_i = I_i + N_i$. Then, the pixel value of the original image is altered by a corresponding noise value. The augmented image I_i now contains a version of the original image with added noise which can reflect variations.

The augmented dataset thus comprises a blend of original and noise-augmented images. By including both types of images, the dataset becomes more representative of various scenarios the model might encounter, including those with imperfect or noisy data. Finally, through this augmented training process, the model is exposed to a broader spectrum of data variability, which is crucial for learning features that are invariant to such noise.

3.2.2 *Salt and Pepper Noise*

In image processing, salt and pepper noise is added to enhance image quality by removing noise and enhancing the robustness of algorithms. Firstly, a noise-free original image is required for processing. Then, an appropriate salt and pepper noise density p which will decide the proportion of pixels in the image that will be affected by noise should be determined. Then, each pixel I(x,y)in the original image will be generated a random number r between 0 and 1 .

For each pixel $I(x, y)$ in the original image:

- If $r < \frac{p}{2}$, set $I(x, y)$ to the minimum pixel value (e.g., 0 for black).
- If $\frac{p}{2} \leq r < p$, set $I(x, y)$ to the maximum pixel value (e.g., 255 for white).
- If $r \geq p$, leave $I(x, y)$ unchanged.

These white and black pixels are introduced to the image and represent the lowest and highest pixel values, respectively, simulating the effect of salt-and-pepper noise.

When removing noise, the median filter is used to remove noise by replacing

each pixel value in the image with the median value of the surrounding pixels while maintaining the images' edges and detail. After applying the filter, the quality of the image is significantly improved with the noise effectively removed, the important details of the image are still intact as shown in the picture.

$$I_{filtered}(x, y) = \text{median}\{I(x', y') \| \ (x', y') \in \text{Neighborhood of}(x, y)\}$$

Where:

- $I_{filtered}(x, y)$ represents the pixel value at position (x, y) in the filtered image.
- $I(x', y')$ is the pixel value in the original image in the neighborhood of (x, y).
- The "Neighborhood of (x, y)" refers to a square window centered at (x, y)

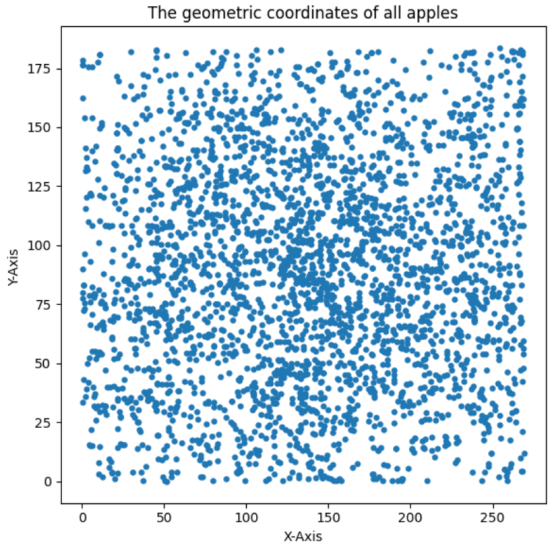


Fig 4. Scatter Diagram of Location of Apples

In the above diagram , the distribution of points seems random, suggesting that the apples are scattered naturally rather than in a controlled manner. The x-coordinates are spread across the entire range of the X-axis, indicating a wide dispersion of apples in this dimension. On the Y-axis, there is a denser gathering of points around the center, hinting at a higher concentration of apples in the central area. The absence of distinct clusters or outliers suggests that there isn't a specific pattern to the apple distribution which could reflect a more organic growth environment, where apples have fallen naturally or have been collected in a non-uniform manner.

3.3 Model Establishment and Solution of Problem 3

3.3.1 *Image Segmentation using GrabCut*

We implemented GrabCut which is an upgraded version of Graphcut.This algorithm utilizes texture information and boundary (contrast) information in images, and can achieve relatively good segmentation results with minimal user interaction.Therefore, we choose GrabCut to segment high quality images due to its following features:

- The GrabCut algorithm utilizes a Gaussian Mixture Model (GMM) to effectively handle the color information across the three RGB channels.
- Segmentation estimation and model parameter learning are performed concurrently which ensures that the model adapts to the input data, resulting in enhanced segmentation accuracy with user guidance.
- Minimal requirement for initial user input and necessitates only a set of pixels representative of the background region to initialize the segmentation process.

3.3.2 *RGB Color Space Model*

We adopt the RGB color space and model the object and background using a full covariance Gaussian Mixture Model (GMM) with K Gaussian components (typically $K = 5$). Thus, there is an additional vector $k = k_1, \ldots, k_n, \ldots, k_N$, where k_n represents which Gaussian component the nth pixel corresponds to, with $k_n 1, ..., K$. For each pixel, it either comes from a Gaussian component of the object GMM or from a Gaussian component of the background GMM.

$$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z)\Pi \tag{1}$$

$$U(\alpha, k, \theta, z) = \sum_{n} (\alpha_n, k_n, \theta_n)\Pi \tag{2}$$

$$D(\alpha_n, k_n, \theta_n, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \sum_{(\alpha_n, k_n)} + \frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^T \sum_{(\alpha_n, k_n)}^{-1} [z_n - \mu(\alpha_n, k_n)]\Pi \tag{3}$$

$$\theta = \pi(\alpha, k), \mu(\alpha, k), \sum_{(\alpha,k)}, \alpha = 0, 1, k = 1 \ldots K\Pi \tag{4}$$

Where, U represents the regional term, which is the same as mentioned in the previous text. It represents the penalty for a pixel being classified as an object or

background, which is the negative logarithm of the probability of a pixel belonging to the object or background. We know that the Gaussian mixture density model is in the following form:

$$D(x) = \sum_{i=1}^{K} \pi_i g(x; mu_i, \Sigma_i), \quad \sum_{=}^{K} \pi_i = 10 \leq \pi_i \leq 1 \tag{5}$$

$$g(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{6}$$

$$V(\alpha, z) = \gamma \sum_{(m,n)\in C} |\alpha_n - \alpha_m| \exp\left(-\beta \|z_m - z_n\|^2\right) \tag{7}$$

3.3.3 *Iterative Energy Minimization Segmentation Algorithm*

We have found that GrabCut is iteratively minimized and with each iteration can improve the parameters of the GMM used for modeling the object and background, resulting in enhancing image segmentation.The workflow of the alogrithm is explained below:

- **Initialization**
    1. **Trimap Creation:** The initiation of the process involved the creation of a 'trimap', a critical step that dictates the initial state of the segmentation process. The pixels that fall outside this frame are collectively acknowledged as background pixels (TB), while the ones enclosed within are regarded as potential candidates for being part of the object, thus termed 'potential object' pixels (TU). For every pixel situated in the TB zone, we assign a label $\alpha_n = 0$ categorizing it unequivocally as a background element. Conversely, each pixel within the TU area is labeled $\alpha_n = 1$, designating it as a potential constituent of the object. Through these initial steps, we attain a preliminary classification—some pixels are identified with the object $\alpha_n = 1$ and others are associated with the background $\alpha_n = 0$.
    2. **Gaussian Mixture Models:** We employed Gaussian Mixture Models (GMMs) to model the color distributions of both the object and the background by u clustering these labeled pixels into K classes via the k-means algorithm. Each class aligns with a Gaussian model within the GMM, a statistical ensemble that captures the essence of our object and background. The parameters of these

models—their means and covariances—are meticulously calculated from the RGB values, infusing the models with the lifeblood of data. The influence of each Gaussian component is then weighed and balanced by the number of pixels it claims.

- **Iterative Minization**

  1. **Gaussian Component:** Assign each pixel to a Gaussian component in the GMM (for example, if pixel n is an object pixel, then substitute pixel n's RGB value into each Gaussian component of the object GMM, and the one with the highest probability is most likely to generate n, i.e., the $k_n - th$ Gaussian component of pixel n).
  2. **GMM Classification:** Optimize the GMM parameters for the given image data Z since in step (1) we have already classified each pixel to a Gaussian component, each Gaussian model now has a set of pixel samples. The mean and covariance of its parameters can be estimated from these pixel samples' RGB values, and the weight of each Gaussian component can be determined by the ratio of the number of pixels belonging to that component to the total number of pixels.
  3. **Graph Cut Classification:** Segmentation estimation (construct a graph using the Gibbs energy terms analyzed in step (1) , establish the weights of t-links and n-links, and then segment using the max flow/min cut algorithm).
  4. **Final Convergence Classification:** Repeat step (1) to (3) until convergence. After the segmentation in step (3) , each pixel's belonging to either the object GMM or the background GMM changes, so each pixel's k_n changes, and therefore the GMM changes too. Thus, each iteration interactively optimizes both the GMM model and the segmentation result. Moreover, since steps (1) to (3) are all processes of energy reduction, it ensures that the iterative process will converge.
  5. **Borader Matting:** Finally, border matting is utilized to smooth the segmented boundaries.

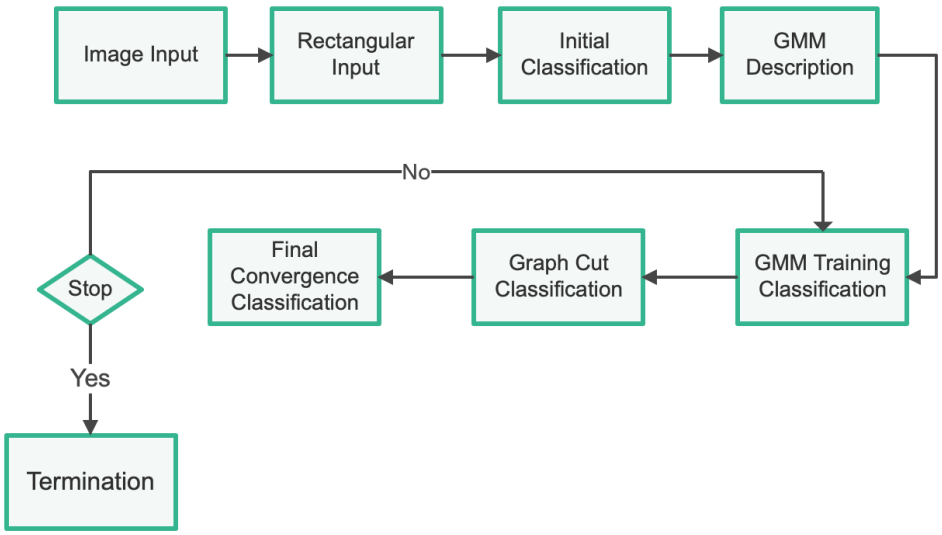


Fig 5. Flowchart of Apple Maturity Assessment Pipeline

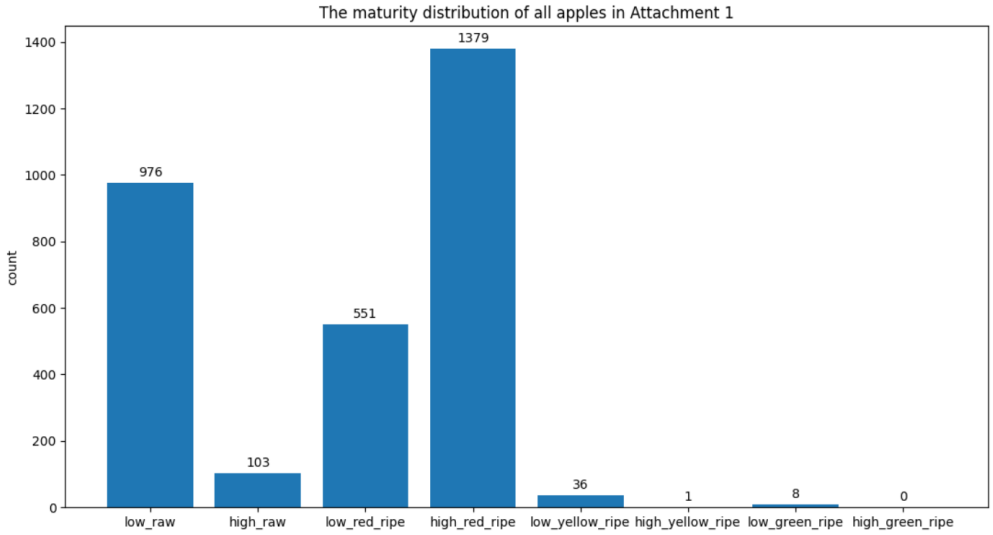


Fig 6. Histogram of Distribution of Apple Maturity Stages

The chart clearly shows that the highest number of apples falls into the 'high red ripe' category, indicating a large proportion of apples are both red and ripe. The 'low raw' category also contains a significant number of apples, suggesting a considerable amount of immature or unripe apples. The other categories have significantly fewer apples. This suggests that in this dataset, ripe red apples are the most common, yellow apples are less common and green apples extremely rare.

3.4 Model Establishment and Solution of Problem 4

3.4.1 *Quantifying Apple Size Through Area Calculations*

In designing our approach for quantifying apples, we decided to employ similar algorithms as used in the previous problem to calculate the masses of apples. Our core process is structured as follows:

Firstly, we begin the foundation of mass estimation by accurately determining the area occupied by each apple in the image. This process starts with isolating the apple from the background, a task we achieve through the GrabCut algorithm, renowned for its effectiveness in segmenting objects in images. Following this segmentation, we employ the K-Means clustering algorithm, focusing on color classification to distinguish between different ripeness levels of the apples. This step is crucial as it directly impacts the subsequent area calculation. We initially compute the area in the standardized 100x100 pixel space and then scale it up proportionally to match the original image dimensions. This scaling is vital to ensure accuracy, as the resized image dimensions are adopted primarily for processing convenience.

Secondly, the core innovation of our method is the conversion of the calculated area into an estimated mass. This is done using a predetermined conversion factor, which in our case is 0.7. This factor is presumably derived from empirical data or established correlations between the visible area of an apple in an image and its actual mass. The choice of this factor is critical as it directly influences the accuracy of the mass estimation. It's a reflection of the density and physical properties of the apple, tailored to the specific varieties and conditions under study.

$$Estimated\ Mass = Area \times Conversion\ Factor \tag{8}$$

where, Conversion Factor = Pre-determined factor for converting area to mass (0.7 in this case)

Finally, the estimated mass is then rounded to three decimal places, a decision that balances precision with practical usability. This level of precision is typically sufficient for agricultural applications, where minor variations in mass are less critical.

Fig 7. Apple Image Processing and Mass Determination

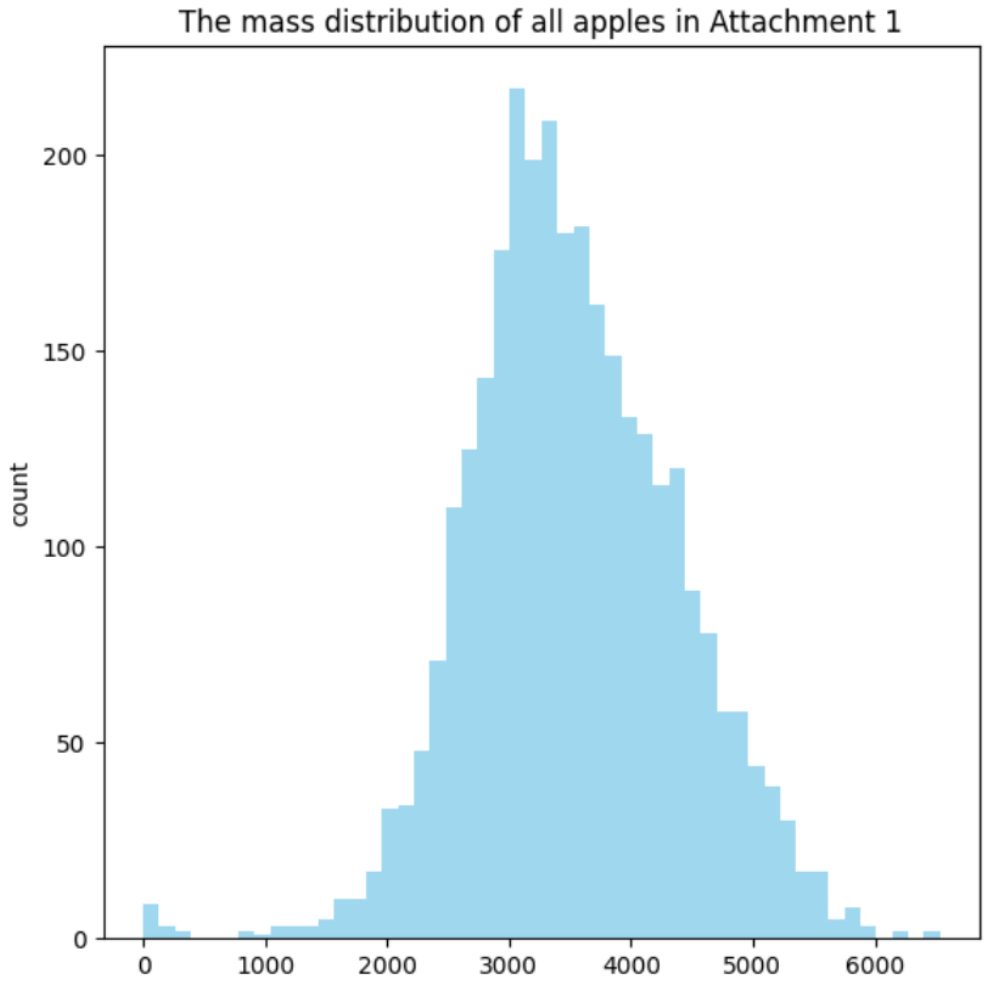


Fig 8. Histogram of Distribution Apple Maturity Stages

The histogram depicts the mass distribution of apples, illustrating a normal distribution centered around 3000 units. The bell-shaped curve suggests that the majority of apples have a mass close to this central value, with fewer apples at the lower and higher ends of the mass spectrum. The tapering tails of the histogram indicate that extremely light or heavy apples are relatively rare in the sample. Overall, the distribution signifies a consistent variability in apple mass, with most apples weighing near the average and progressively fewer as the weight increases or decreases from this central peak

3.5 Model Establishment and Solution of Problem 5

3.5.1 *The Recognition of Apples and Fruit Classification*

This model is designed for object classification, ConvNet aims to extract features from images and classify them into predefined categories. The architecture comprises three convolutional layers, followed by max pooling and two fully connected layers.

3.5.2 *ConvNet*

Convolutional Layers. We design three convolutional layers, forming the fundamental building blocks of a CNN. Each layer performs convolution operations on the input data using filters (or kernels). The parameters for the model, including the number of input channels (3 for RGB images), the size of the filters in each layer, and also the padding which are necessary for the model architecture. Each filter extracts different features from the input and each padding ensures that the spatial dimensions are either maintained or appropriately reduced. The size of the filter set to $filter\ size\ //\ 2$ to maintain the size of the output feature map if the stride is 1. (where the $filter\ size$ is set to 3 for all the filters)

Pooling Layer. We design max pooling layers to reduce the spatial dimensions (width and height) of the input volume for the next convolutional layer. The pooling size is set to 2×2, which reduces the size of the input features by half in both dimensions. Stride is also set to 2, which is the step the pooling window takes while moving across the input.

Fully Connected Layers. The two linear layers are used for which the first linear network connects the flattened output of the last convolutional layer to the first fully connected layer with 128 neurons. The second linear layer connects the first fully connected layer to the output layer with 5 neurons, corresponding to the number of classes for classification.

Forward Pass. The forward method defines how the input data x flows through the network. The input is first converted to float and moved to the appropriate device (GPU). It is then permuted to match the expected format $[batch\ size, channels, height, width]$. Input passes sequentially through the three convolutional layer, with ReLU activation and max pooling applied after each convolution. The output is flattened before being passed to the fully connected layers. Finally, it passes through the first fully connected layer with a ReLU activation and then through the second fully connected layer to produce the final output.

Auxiliary Methods. We calculate the flattened size of the output after the last convolutional layer. This is crucial for connecting the convolutional layers to the fully connected layers. The model then simulates the forward pass of the convolutional part of the network with a random input to compute the output size for the fully connected layer.

This CNN model, with its convolutional, pooling, and fully connected layers,

is typical for image classification tasks. The convolutional layers extract features from the images, pooling layers reduce the dimensionality, and fully connected layers make the final classification decision based on these features.

3.5.3 *Hyperparameters and Configurations*

1. **Size and Number of the Filters:** 3 layers with same filter sizes 3×3 and varying numbers of filters (32, 32, 64) respectively.
2. **Number of the Neurons:** 128 neurons in one hidden layer.
3. **Size of the Input Images:** 180×270 pixels with 3 color channels.
4. **Classes:** 5 distinct classes for fruit classification.
5. **Batch Size:** 32, indicating the number of samples processed before internal parameters of the model are updated.
6. **Size of the Dataset:** 20,705 images.
7. **Training/Test Split:** 80% training, 20% testing.
8. **Learning Rate and Weight Decay:** Parameters for the optimizer to control the learning process and regularize the model.
9. **Optimizer:** Adam
10. **Loss Function:** Cross Entropy Loss

3.5.4 *Training Loop*

Overview of the Training Loop. The training loop is designed to iteratively update the model's parameters (weights) based on the training data. It involves several key steps:

- **Initializing Performance Metrics:** Accumulators for True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN), and other statistics are set up for calculating accuracy, precision, recall, and F1 score.
- **Iterative Training Over Epochs:** The model is trained over a number of epochs, where each epoch represents a complete pass through the entire training dataset.
- **Batch Processing:** In each epoch, the training data is processed in batches. This is crucial for efficient memory usage and effective gradient updates.
- **Forward Pass and Loss Computation:** For each batch, a forward pass (predicting labels using the current model state) is conducted, followed by loss computation using Cross Entropy Loss.

- **Backward Pass and Parameter Update:** The loss is then used to perform a backward pass to compute gradients, which are subsequently used to update the model's parameters using the Adam optimizer.
- **Performance Evaluation:** After each epoch, the model's performance is evaluated using metrics like accuracy, precision, recall, and F1 score, both on training and test datasets.
- **Model Checkpointing:** The best-performing model state is saved for future use or further analysis.

IV. Model Evaluation and Improvement

4.1 Model Evaluation of Problem 1

Table 1 Comparison of MAE and MSE before and after Augmentation

Metric	Before Augmentation	After Augmentation
MAE	14.39	11.67
MSE	468.59	392.00

The reduction in both MAE and MSE in the context of counting apples suggests that the data augmentation process enriched the training dataset with a greater variety of apple images or scenarios, leading to more precise apple count predictions. This improvement indicates that data augmentation was successful in exposing the model to a wider range of apple-related features, which in turn improved the model's ability to generalize and accurately count apples in varied situations.

4.2 Model Evaluation of Problem 5

Key Formulas and Metrics.

Accuracy: The proportion of correctly predicted observations to the total observations.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{9}$$

Precision: The ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Recall (Sensitivity): The ratio of correctly predicted positive observations to the all observations in actual class.

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

F1 Score: The weighted average of Precision and Recall.

$$F1\ Score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

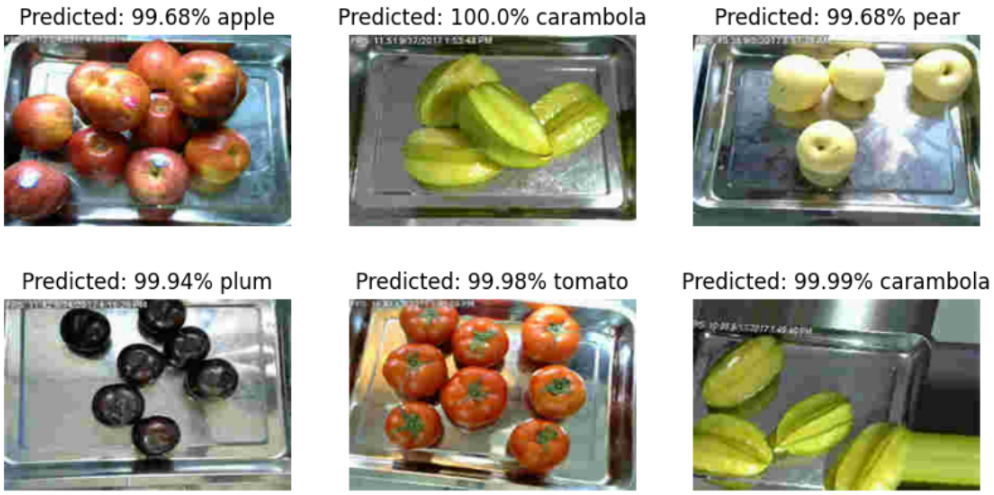


Fig 9. Prediction of Fruit Classification

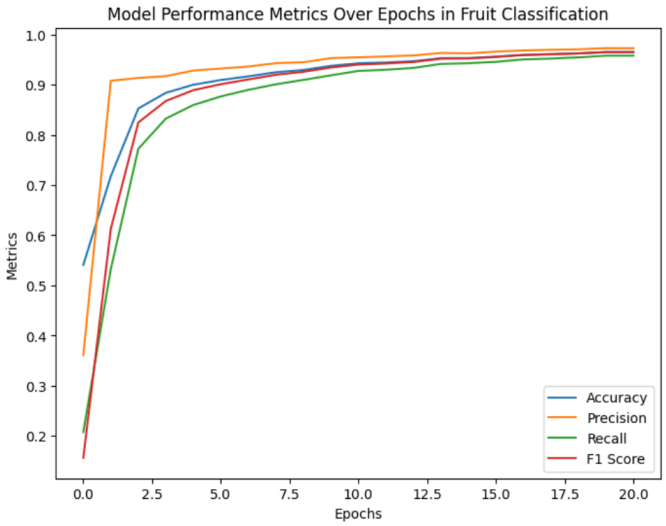
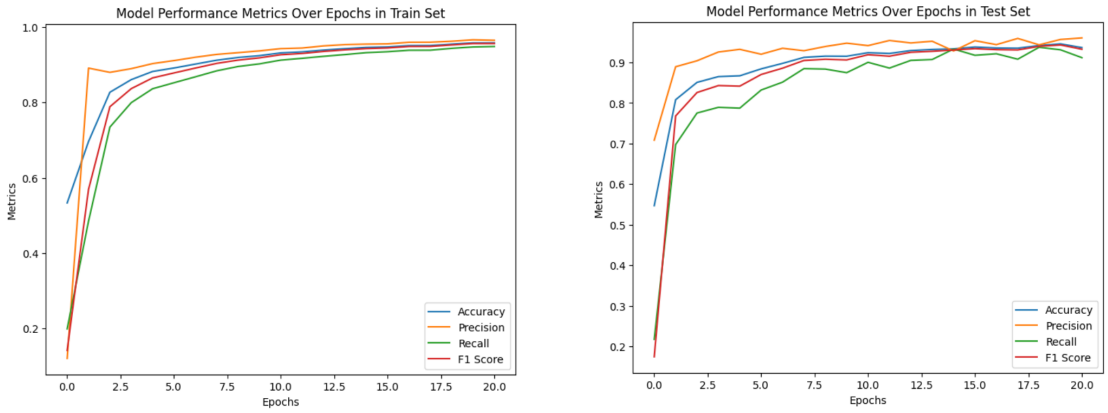


Fig 10. Model Performance Metrics Over Epochs In Fruit Classification



(a) Training Performance **(b) Test Performance**

Fig 11. Model Performance Metrics Over Epochs In Fruit Classification

Table 2 Model Evaluation on Train Set and Test Set

Dataset	Accuracy	Precision	Recall	F1-Score
Train	95.81%	96.49%	94.85%	95.64%
Test	93.65%	96.03%	91.17%	93.25%

The table presents a model evaluation with robust performance metrics on both training and test datasets. On the training set, the model demonstrates higher percantages than the train model. Despite the slight decline, the model's performance on the test set remains strong, suggesting that it generalizes well to new data and maintains a balanced trade-off between precision and recall. This implies that the model is reliable and could be effectively used for further predictions or in practical applications where similar data is encountered.

4.2.1 *Strength and Weakness*

Strength: The model is skilled at recognizing various features of apples, adapting to new data through its training process, and showing signs of strong predictive power. This adaptability is crucial for applications where a wide range of apple characteristics must be accurately identified, such as in sorting different apple varieties or assessing their quality. Moreover, the model's effectiveness is bolstered by its exposure to a diverse set of examples, which equips it to handle a broad spectrum of real-world scenarios reliably.

Weakness: The model does exhibit certain limitations that could hinder its practical deployment. The most significant of these is the discrepancy in performance when transitioning from familiar training data to unfamiliar test data, suggesting that the model might struggle with completely new inputs or underrepresented apple features. This discrepancy could lead to less reliable predictions in a commercial setting where the stakes for accuracy are high, such as in precise yield predictions or during the grading process for market-ready apples.

V. References

[1] C. Rother, V. Kolmogorov, A. Blake, *GrabCut: Interactive foreground extraction using iterated graph cuts*[J], ACM Transactions on Graphics, 2004, 23, 309-314.

[2] N. Häni, P. Roy, V. Isler, *Apple counting using convolutional neural networks*, In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, October, pp. 2559-2565.

[3] A. van Meekeren, M. Aghaei, K. Dijkstra, *Exploring the Effectiveness of Dataset Synthesis: An application of Apple Detection in Orchards*, arXiv preprint arXiv:2306.11763, 2023.

[4] rdcolema, *CNN for multi-class image recognition in tensorflow*, GitHub, About. Visited on (2023, 11, 23). `https://github.com/rdcolema/tensorflow-image-classification`

[5] Tulika Choudhary, *Salt-PepperNoiseRemoval-Cpp*, GitHub. Visited on (2023, 11, 24). `https://github.com/cvillacampa/GPInputNoise`

VI. Appendix

Listing 1: Convulutional Neural Network of Objection Recognition(Q5)

```
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        # Convolutional layers
        self.conv1 = nn.Conv2d(C_IMG, FILTER_NUM_1, FILTER_SIZE_1,
            padding=FILTER_SIZE_1//2)
        self.conv2 = nn.Conv2d(FILTER_NUM_1, FILTER_NUM_2,
            FILTER_SIZE_2, padding=FILTER_SIZE_2//2)
        self.conv3 = nn.Conv2d(FILTER_NUM_2, FILTER_NUM_3,
            FILTER_SIZE_3, padding=FILTER_SIZE_3//2)

        # Pooling layer
        self.pool = nn.MaxPool2d(2, 2)

        # Fully connected layers
        # Calculate the size of the flattened layer
        self.fc1_size = self._get_conv_output((C_IMG, H_IMG, W_IMG))
        self.fc1 = nn.Linear(self.fc1_size, FC_SIZE)
        self.fc2 = nn.Linear(FC_SIZE, NUM_CLASSES)

    def forward(self, x):
        x = x.float().to(device)

        # Reorder to [BATCH_SIZE, channels, height, width]
        x = x.permute(0, 3, 1, 2)

        # Apply convolutions and max pooling
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))

        # Flatten the output for the dense layer
        x = x.reshape(-1, self.fc1_size)
```

```
        # Fully connected layers
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

    def _get_conv_output(self, shape):
        with torch.no_grad():
            input = torch.rand(1, *shape)
            output = self._forward_features(input)
            return int(np.prod(output.size()))

    def _forward_features(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        return x

model = ConvNet().to(device)
print(model)
```

Listing 2: GrabCut and Classifaction of Apples(Q4)

```
# grabCut
def GrabCut(img):
    mask = np.zeros(img.shape[:2],np.uint8)

    bgdModel = np.zeros((1,65), np.float64)
    fgdModel = np.zeros((1,65), np.float64)
    rect = (0, 0, img.shape[1]-1,img.shape[0]-1)
    iterCount = 5
    cv2.grabCut(img, mask, rect, bgdModel, fgdModel,
        iterCount,cv2.GC_INIT_WITH_RECT)
    mask2 = np.where((mask==2) | (mask==0), 0, 1).astype('uint8')
    img = img * mask2[:,:,np.newaxis]
    return img

# Maturity and Ripeness of Apples (Q3,Q4)
```

```
def question3(json_file, image_file):
    Q_3 = []
    with open(json_file,'r') as file:
        data_label = json.load(file)
    data_img = cv2.imread(image_file)
    data_img = data_img[:,:,::-1]
    for circle in data_label:
        # Crop Image
        crop_img = cropImg(data_img, circle)

        # Resizing
        crop_img =
            cv2.resize(crop_img,(100,100),fx=0.8,fy=0.8,interpolation=2)

        # Grabcut
        grabcut_img = GrabCut(crop_img)

        # KMeans for considering Ripness
        kMeans_img = kmeans(k=5, img=grabcut_img)
        color = meanExceptBlack(kMeans_img)
        Ripeness = IdentifyRipeness(color)
        Q_3.append(Ripeness)

    return Q_3
```