NIST UNIVERSITY

ACADEMIC YEAR : 2025-26

SUBJECT : DESIGN ANALYSIS AND ALGORITHM

CODE : CSE200T

SECTION : D

SEMESTER : 3RD

NAME OF THE STUDENT	ROLL NUMBER
SHREYA BADATYA	202457463
SIMON KUMAR PANDA	202456424

TOPIC:

AUTOMATIC FRUIT QUALITY SORTING

Github Link : https://github.com/shreya2007-del

Github Link : https://github.com/simon-2005

Name of the Faculty: PROF. SWATIKANTA MISHRA

Comment:

Marks Obtained:

TITLE :

Design and Implementation of Automatic Fruit Quality Sorting using OpenCV Image Processing.

ABSTRACT :

Fruit grading is an essential process in agriculture, but manual sorting is often slow, inconsistent, and subjective. Variations in human judgment make it difficult to accurately evaluate important fruit quality factors such as ripeness, color, and shape. As a result, farmers may face financial losses due to inaccurate or inefficient sorting methods. To address these challenges, automated fruit quality assessment has become an effective alternative.

This project presents an Automatic Fruit Quality Sorting System developed using Python**,** OpenCV, and NumPy. The system analyzes fruit images to extract features like color saturation and circularity, which indicate ripeness and shape quality. Based on these features, a quality score is generated for each fruit, and the fruits are sorted using the Merge Sort algorithm. This automated approach provides faster, more reliable, and more consistent fruit grading, making it suitable for farmers, vendors, and small-scale agricultural industries.

To efficiently arrange the fruits from highest to lowest quality, the Merge Sort algorithm is implemented. This ensures fast and stable sorting even when dealing with larger datasets. The system is designed to be simple, lightweight, and accessible, making it suitable for farmers, small vendors, and agricultural collection centers.

By integrating computer vision with classical sorting techniques, this project demonstrates a practical and cost-effective approach to fruit quality evaluation. It highlights how digital tools can support rural farming communities, improve market readiness, and reduce losses caused by inaccurate manual grading.

INTRODUCTION:

In modern agriculture, fruit quality plays a crucial role in determining market value, storage decisions, and consumer acceptance. However, manual fruit sorting remains the most commonly used method in many regions, especially in rural markets. Manual grading depends heavily on human experience, which often leads to inconsistent results and slow processing, especially when handling large quantities of produce.

With advancements in image processing and machine learning, automated systems have become a practical solution for improving sorting accuracy. This project focuses on building an Automatic Fruit Quality Sorting System using Python, OpenCV, and NumPy. The system evaluates fruits based on measurable image features such as color, ripeness, and shape to produce an objective quality score. The fruits are then sorted using the Merge Sort algorithm for efficient ranking. The aim is to create a simple yet effective tool that supports farmers by reducing sorting time, minimizing errors, and improving overall fruit quality assessment.

Beyond improving accuracy, automated fruit quality assessment also helps standardize the grading process across different markets and suppliers. In traditional systems, the same batch of fruits may receive different quality ratings depending on who evaluates them. By using computer vision–based methods, the grading procedure becomes uniform and repeatable. This ensures fair pricing for farmers and consistent quality for consumers. Moreover, such systems can be easily integrated with conveyor belts or packaging units, enabling semi-automatic or fully automated sorting in small agricultural setups. As technology becomes more accessible, these solutions offer an affordable and scalable approach for modernizing fruit quality evaluation.

OBJECTIVES:

1. To develop an automated fruit quality evaluation system using Python and OpenCV.
2. To analyze fruit ripeness based on color and saturation in HSV space.
3. To detect fruit shape and defects using contour extraction and circularity measurement.
4. To compute an overall fruit quality score from measurable image features.
5. To sort fruits from best to worst quality using the Merge Sort algorithm.
6. To provide a fast, consistent, and low-cost alternative to manual fruit grading.
7. To assist farmers and vendors with a reliable system for improving sorting accuracy.

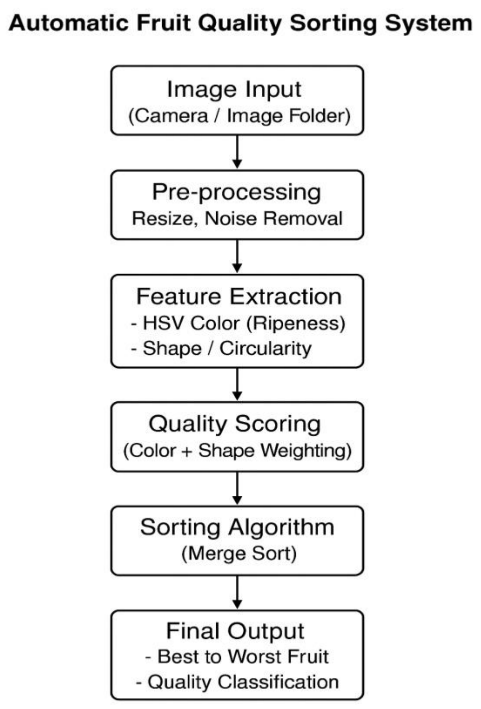
PROBLEM DEFINITION :

Manual fruit sorting in farms and markets is often slow, inconsistent, and highly dependent on human judgment, leading to errors in assessing ripeness, color, and shape. This results in poor grading accuracy and economic loss for farmers. To address this issue, there is a need for an automated system that can evaluate fruit quality objectively and efficiently. This project aims to develop an image-processing–based fruit quality sorting system using Python, OpenCV, and NumPy to analyze fruit characteristics such as color saturation and shape circularity, generate a quality score, and sort fruits accurately using the Merge Sort algorithm.

SCOPE :

The scope of this project includes the development of an automated fruit quality grading system using Python, OpenCV, and NumPy. The system is capable of analyzing fruit images to evaluate key quality parameters such as color, ripeness, and shape, and then assigning a quality score. It further sorts fruits from best to worst using the Merge Sort algorithm. The project is limited to image-based analysis and can classify only one fruit per image. It can be extended to support multiple fruits, real-time camera input, and integration with mechanical sorting machines for large-scale agricultural use.

DIAGRAM :



ALGORITHM :

Step 1: Start

Step 2: Import required libraries

- Import OpenCV (cv2)
- Import NumPy (numpy)

Step 3: Define Merge Sort algorithm

- Recursively divide the list of fruits into two halves.
- Sort each half.
- Merge the halves in descending order of quality score.

Step 4: Define quality classification function

- If score ≥ 70 → Best Quality
- If score ≥ 50 → Good Quality
- If score ≥ 30 → Average Quality
- Else → Poor Quality

Step 5: Define function to calculate fruit quality

1. Read the image.
2. Resize image to fixed size.
3. Convert to HSV color space.
4. Extract mean saturation (ripeness indicator).
5. Convert to grayscale.
6. Apply Otsu thresholding to segment the fruit.
7. Detect external contours.
8. Select the largest contour (fruit region).
9. Compute:
    - Area
    - Perimeter
    - Circularity = $(4\pi \times \text{Area}) / \text{Perimeter}^2$
10. Calculate Final Score = Color Weight + Shape Weight.

Step 6: Load fruit images

- Store filenames in a list.
- For each image:
    - Calculate quality score.
    - Append (name, score) to list.

Step 7: Sort the fruits using Merge Sort

1. Sort from highest score to lowest score.

Step 8: Display output

- Print fruit name, score, and quality label (Best / Good / Average / Poor).

Step 9: End

<u>PSEUDOCODE</u> :

```
BEGIN

   DEFINE function merge_sort(arr)

      IF length of arr > 1 THEN

         mid ← length(arr) / 2

         L ← left half of arr

         R ← right half of arr

         merge_sort(L)
```

```
        merge_sort(R)
        i, j, k ← 0
        WHILE i < length(L) AND j < length(R) DO
            IF L[i].score > R[j].score THEN
                arr[k] ← L[i]
                i ← i + 1
            ELSE
                arr[k] ← R[j]
                j ← j + 1
            ENDIF
            k ← k + 1
        ENDWHILE
        WHILE i < length(L) DO
            arr[k] ← L[i]
            i ← i + 1
            k ← k + 1
        ENDWHILE
        WHILE j < length(R) DO
            arr[k] ← R[j]
            j ← j + 1
            k ← k + 1
        ENDWHILE
    ENDIF
END FUNCTION
DEFINE function quality_message(score)
```

```
    IF score ≥ 70 THEN RETURN "BEST QUALITY"
    ELSE IF score ≥ 50 THEN RETURN "GOOD QUALITY"
    ELSE IF score ≥ 30 THEN RETURN "AVERAGE QUALITY"
    ELSE RETURN "POOR QUALITY"
END FUNCTION
DEFINE function get_quality_score(image_path)
    LOAD image from image_path
    IF image NOT FOUND THEN
        PRINT "Image not found"
        RETURN 0
    ENDIF
    RESIZE image to $(300 \times 300)$
    CONVERT image to HSV color space
    mean_saturation ← average of saturation channel
    CONVERT image to grayscale
    APPLY Otsu thresholding to segment fruit
    FIND contours in thresholded image
    IF no contours found THEN
        RETURN 0
    ENDIF
    largest_contour ← contour with maximum area
    area ← contour area
    perimeter ← contour perimeter
    IF perimeter = 0 THEN circularity ← 0
    ELSE circularity ← $(4 \times \pi \times \text{area}) / (\text{perimeter}^2)$
```

```
        ENDIF
        score ← (mean_saturation / 255 × 40) + (circularity × 60)
        RETURN score
    END FUNCTION
    fruit_list ← ["apple1.jpg", "apple2.jpg", "apple3.jpg", "apple4.jpg"]
    fruits ← empty list
    FOR each image IN fruit_list DO
        score ← get_quality_score(image)
        ADD (image, score) TO fruits
    ENDFOR
    CALL merge_sort(fruits)
    PRINT "SORTED FRUITS (BEST TO WORST)"
    FOR each (image, score) IN fruits DO
        PRINT image, score, quality_message(score)
    ENDFOR
END
```

TIME COMPLEXITY :

1. Image Processing (OpenCV):
Each fruit image is processed once to extract color, shape, and size features.

➡**Time Complexity: O(n)** per image.

2. Merge Sort for Sorting the Fruits:
Merge Sort divides and merges the list of fruits.

➡**Time Complexity: O(n log n)**

3. Overall Time Complexity:

Since image processing (O(n)) and sorting (O(n log n)) occur together, the total complexity is:

➡ **O(n log n)**

- **SPACE COMPLEXITY :** O(n)

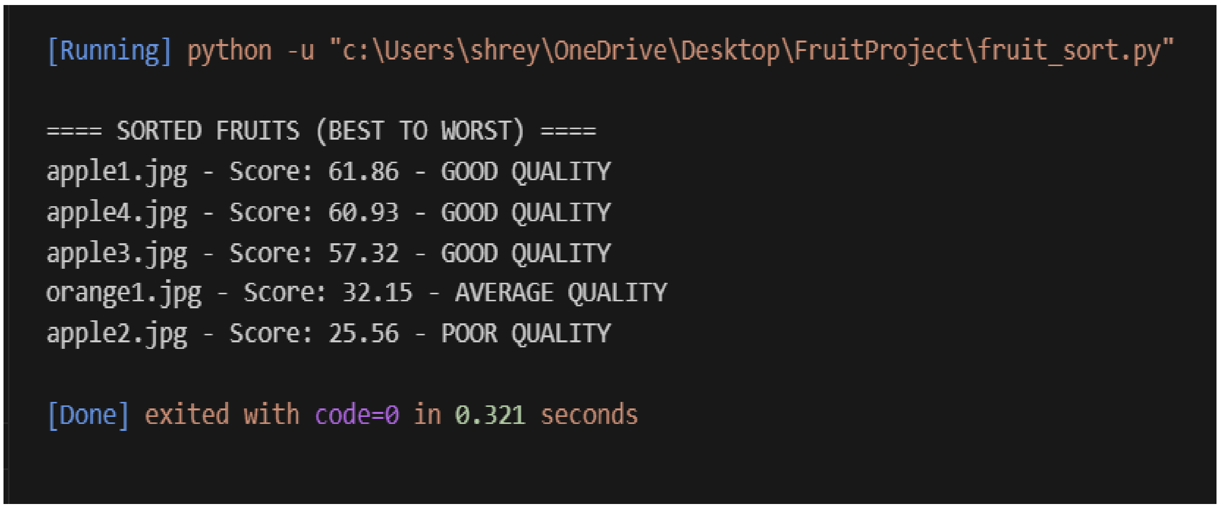
CODE :

Github link -1: https://github.com/shreya2007-del/Fruit-Quality-Sorting.git

Github link-2: https://github.com/Simon-2005/Fruit-Quality-Sorting-OpenCV.git

The GitHub link contains all program files, images, and documentation used in this project.

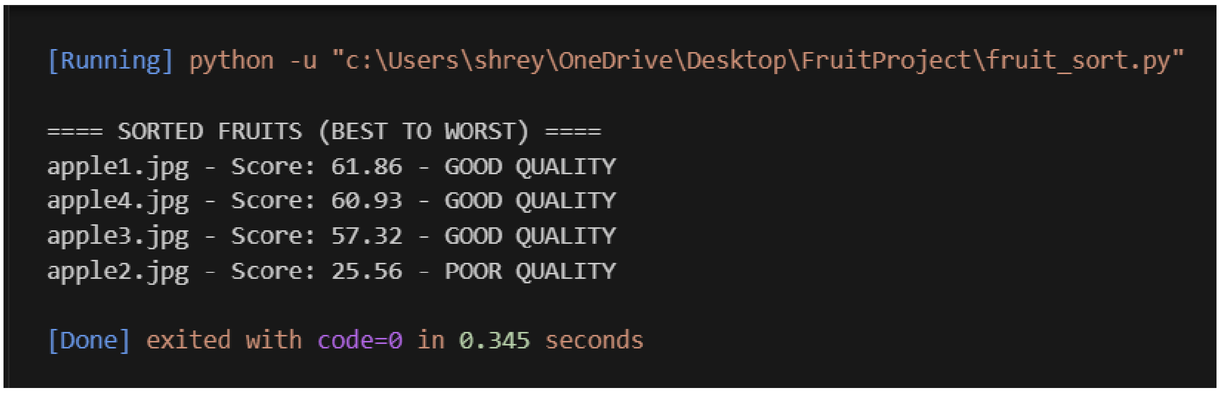
OUTPUT :



```
[Running] python -u "c:\Users\shrey\OneDrive\Desktop\FruitProject\fruit_sort.py"

==== SORTED FRUITS (BEST TO WORST) ====
apple1.jpg - Score: 61.86 - GOOD QUALITY
apple4.jpg - Score: 60.93 - GOOD QUALITY
apple3.jpg - Score: 57.32 - GOOD QUALITY
orange1.jpg - Score: 32.15 - AVERAGE QUALITY
apple2.jpg - Score: 25.56 - POOR QUALITY

[Done] exited with code=0 in 0.321 seconds
```



```
[Running] python -u "c:\Users\shrey\OneDrive\Desktop\FruitProject\fruit_sort.py"

==== SORTED FRUITS (BEST TO WORST) ====
apple1.jpg - Score: 61.86 - GOOD QUALITY
apple4.jpg - Score: 60.93 - GOOD QUALITY
apple3.jpg - Score: 57.32 - GOOD QUALITY
apple2.jpg - Score: 25.56 - POOR QUALITY

[Done] exited with code=0 in 0.345 seconds
```

CONCLUSION :

The Automatic Fruit Quality Sorting System developed in this project successfully demonstrates how computer vision and algorithmic processing can be used to automate one of the most labor-intensive tasks in agriculture: fruit grading. Traditional fruit sorting relies heavily on human judgment, which is often affected by fatigue, inconsistency, and subjective evaluation. This project addresses these challenges by providing a fast, objective, and reliable method of assessing fruit quality using Python, OpenCV, and NumPy.

By extracting key features such as color saturation (to estimate ripeness) and circularity (to measure fruit shape and detect deformities), the system generates a measurable quality score for each fruit image. This score provides an unbiased assessment based on quantifiable parameters rather than visual guessing. The use of the Merge Sort algorithm ensures that fruits are arranged efficiently from best to worst quality, maintaining accuracy even when the number of fruits increases.

The results clearly show that computer vision can significantly improve sorting consistency compared to manual methods. The system's modular design allows it to be easily extended to other fruits, additional quality parameters, or real-time applications using cameras and conveyor systems. It can also be integrated into mobile or low-cost embedded platforms, making it practical for small farmers and local markets.

In conclusion, this project proves that fruit quality assessment can be automated effectively using image processing. It not only enhances accuracy but also reduces labor, speeds up the sorting process, and supports farmers in achieving better pricing and reduced wastage. With further refinement—such as texture analysis, color uniformity detection, and machine learning-based classification—the system can evolve into a fully scalable and intelligent agricultural tool suitable for large-scale commercial use.