
# CDS6334
# Visual Information Processing
# Assignment

## Lab Section
## TT2L

## Student Name

Lee Tian Xin
1211301744

# Table Content

# Abstract

This work presents an algorithm for automatic segmentation of fruits and rotten regions in images. The objective is to develop a method that can accurately identify these areas based on a provided dataset of 80 images with corresponding ground truth segmentation. The proposed approach utilizes K-Means clustering to group pixels based on their color features. By analyzing the average colors within each cluster, the algorithm aims to differentiate between fruit, rotten, and potentially background regions. The analysis reveals challenges related to lighting variations, background color inconsistencies, and white apples. Future work could involve pre-processing techniques, feature integration and machine learning techniques.

# Introduction

Automatic segmentation of fruits and rotten regions in images plays a crucial role in various applications, including: Smart agriculture: Identifying healthy and rotten fruits can automate sorting and quality control processes in farms and packing houses. Food safety inspection: Automatic segmentation can assist in detecting spoilage and ensuring food quality during inspection. Robotic harvesting: Segmentation can guide robots to selectively pick ripe fruits and avoid damaged ones. This work addresses this challenge by developing an algorithm based on K-Means clustering to automatically segment fruits and rotten regions in images.

# Description of Methods

The segmentImage function automatically segments fruits and rotten areas in images. It uses techniques including Gaussian blur, K-means clustering, Masking, Morphological Operations.

## 1. Preprocessing

Gaussian Blur: The function starts with applying a Gaussian blur (cv2.GaussianBlur) to the input image (img). This aims to reduce noise and smooth out color variations, potentially improving the clustering results.

Color Space Conversion: The image is then converted from BGR (OpenCV's default color format) to RGB format (cv2.COLOR_BGR2RGB).

## 2. Feature Extraction and K-Means Clustering

2.1. Feature Extraction

For K-Means analysis, the image is reshaped into a single feature vector (vectorized_img) where each row represents a pixel's RGB values in floating-point format (np.float32). This conversion improves K-Means efficiency and allows it to effectively analyze the color information of each pixel

2.2. K-Means Clustering:

The K-Means algorithm (cv2.kmeans) is applied to the reshaped and converted data.
Here's what the parameters signify:

- Vectorized_img_float : The input data
- Number of clusters (K=3) : Identify fruit, rotten, background.
- None : No initial cluster centers provided.
- Criteria : This defines the maximum number of iterations (10) and a minimum improvement threshold for stopping the K-Means process.
- 10 : Maximum number of iterations allowed.
- cv2.KMEANS_PP_CENTERS: Initialization method (PlusPlus for K-Means++).

The output of K-Means includes:

- ret: Flag indicating successful execution.
- label: An array assigning each pixel a cluster label (0, 1, or 2).
- center: The average color (RGB values) for each identified cluster.

## 3. Segmentation Mask Generation

Red-Blue Difference Calculation: It calculates the difference between the Red (R) and Blue (B) values for each cluster center.

Cluster Sorting: Based on the calculated R-B differences, the function sorts the clusters. This sorting helps us identify potential fruit, rotten, and background regions based on the assumption that:

- Fruit: Generally have higher Red (R) compared to Blue (B), resulting in a high R-B difference.
- Rotten: May appear more grayish or bluish, leading to a lower or potentially negative R-B difference.
- Background: The sorting might reveal a cluster with the lowest R-B difference, potentially corresponding to the background.

Based on the cluster analysis, a mask image (mask) is created with the same dimensions as the original image. Each pixel in the mask is assigned a value based on its corresponding cluster label from K-Means:

- 0: Background (potentially the cluster with the lowest R-B difference)
- 1: Rotten (cluster with a low or negative R-B difference)
- 2: Fruit (cluster with a high R-B difference)

Morphological Operations: The mask undergoes morphological closing (to fill small holes) and opening (to remove noise) operations using kernels (np.ones) for smoothening and improving the segmentation results.

## 4. Output

The final function returns the generated segmentation mask (mask), which represents the image with each pixel classified as background, rotten, or fruit based on the K-Means clustering and color analysis.

# Results & Analysis

### Overview

| Segment | Error | Precision | Recall | IoU |
|---|---|---|---|---|
| Background | 0.0745 | 0.9277 | 0.9312 | 0.8733 |
| Rot | 0.3137 | 0.714 | 0.7158 | 0.5831 |
| Fruit | 0.185 | 0.8248 | 0.8343 | 0.7237 |
| | | | | |
| All | 0.1911 | 0.8222 | 0.8271 | 0.7267 |

Figure 1: Result of Segmentation

Figure 1 summarizes the performance of an image segmentation algorithm that classifies pixels into three categories: background, rotten, and fruit. Error: This value (0.0745 - 0.3137) represents the error rate for each segment. Precision: This value (0.714 - 0.9277) represents the proportion of pixels correctly classified as a specific class.Recall: This value (0.7158-0.9312) represents the proportion of actual pixels in a given class that were correctly identified by the algorithm. IoU (Intersection over Union): This value (0.5831 - 0.8733) represents the area of overlap between the predicted segmentation mask and the ground truth mask (actual labels for each pixel) for a specific class.

From the table, we can see that the background class has the lowest error rate (0.0745), highest precision (0.9277), and highest IoU (0.8733), indicating the algorithm performs well in segmenting the background. The fruit class also achieves good performance with a moderate error rate (0.185), high precision (0.8248), and a reasonable IoU (0.7237). Rotten: The rotten class has the highest error rate (0.3137), lower precision (0.714), and the lowest IoU (0.5831) compared to the other classes. This suggests that the algorithm might have some difficulty segmenting rotten regions accurately.

Overall, the segmentation performed well with 0.1911, precision 0.8222 and recall 0.8271.

| Image | Error | Precision | Recall | IoU |
|---|---|---|---|---|
| 1 | 0.1539 | 0.8361 | 0.879 | 0.7516 |
| 2 | 0.045 | 0.9519 | 0.9586 | 0.9139 |
| 3 | 0.1336 | 0.9317 | 0.8357 | 0.7775 |
| 4 | 0.4855 | 0.6121 | 0.4684 | 0.4504 |
| 5 | 0.1675 | 0.8111 | 0.8973 | 0.7286 |
| 6 | 0.3656 | 0.6745 | 0.8845 | 0.5652 |
| 7 | 0.0502 | 0.9539 | 0.947 | 0.905 |
| 8 | 0.1619 | 0.8938 | 0.8128 | 0.7387 |
| 9 | 0.4361 | 0.5778 | 0.5519 | 0.487 |
| 10 | 0.0301 | 0.9699 | 0.97 | 0.9417 |
| 11 | 0.0508 | 0.9466 | 0.9537 | 0.9033 |
| 12 | 0.0368 | 0.9641 | 0.9629 | 0.9293 |
| 13 | 0.0498 | 0.9481 | 0.9547 | 0.9053 |
| 14 | 0.1584 | 0.8099 | 0.922 | 0.741 |
| 15 | 0.2763 | 0.7218 | 0.8657 | 0.6025 |
| 16 | 0.6681 | 0.3462 | 0.3195 | 0.3125 |
| 17 | 0.1707 | 0.847 | 0.8365 | 0.7166 |
| 18 | 0.2595 | 0.7416 | 0.7398 | 0.6313 |
| 19 | 0.1034 | 0.9033 | 0.8939 | 0.8175 |
| 20 | 0.022 | 0.9796 | 0.977 | 0.9572 |
| 21 | 0.0447 | 0.9388 | 0.975 | 0.9159 |
| 22 | 0.134 | 0.8613 | 0.8999 | 0.7762 |
| 23 | 0.0468 | 0.9634 | 0.9474 | 0.9119 |
| 24 | 0.0733 | 0.9297 | 0.9287 | 0.8663 |
| 25 | 0.0811 | 0.9096 | 0.9308 | 0.8507 |
| 26 | 0.0517 | 0.9506 | 0.9473 | 0.9024 |
| 27 | 0.1154 | 0.8818 | 0.8885 | 0.7987 |
| 28 | 0.6196 | 0.4237 | 0.3914 | 0.3349 |
| 29 | 0.1738 | 0.8401 | 0.8677 | 0.72 |
| 30 | 0.4598 | 0.5492 | 0.541 | 0.4477 |
| 31 | 0.395 | 0.602 | 0.631 | 0.5322 |
| 32 | 0.2262 | 0.775 | 0.8271 | 0.661 |
| 33 | 0.1947 | 0.8094 | 0.8272 | 0.6919 |
| 34 | 0.076 | 0.9289 | 0.9225 | 0.8599 |
| 35 | 0.6254 | 0.3914 | 0.3739 | 0.3401 |
| 36 | 0.1151 | 0.8926 | 0.8899 | 0.7996 |
| 37 | 0.1156 | 0.916 | 0.8701 | 0.803 |
| 38 | 0.0589 | 0.9377 | 0.9475 | 0.8904 |
| 39 | 0.0358 | 0.9694 | 0.9601 | 0.9314 |
| 40 | 0.1096 | 0.8924 | 0.8893 | 0.8099 |
| 41 | 0.0596 | 0.9486 | 0.9361 | 0.8881 |
| 42 | 0.0983 | 0.9027 | 0.9013 | 0.8237 |
| 43 | 0.1832 | 0.7852 | 0.8929 | 0.748 |
| 44 | 0.1633 | 0.852 | 0.8404 | 0.7208 |
| 45 | 0.1304 | 0.9045 | 0.8719 | 0.78 |
| 46 | 0.1843 | 0.8247 | 0.812 | 0.7044 |
| 47 | 0.5733 | 0.455 | 0.4029 | 0.3636 |
| 48 | 0.0651 | 0.9343 | 0.9397 | 0.879 |
| 49 | 0.1148 | 0.8844 | 0.8885 | 0.7987 |
| 50 | 0.1402 | 0.8903 | 0.8536 | 0.7618 |
| 51 | 0.1888 | 0.807 | 0.8312 | 0.6869 |
| 52 | 0.2364 | 0.7988 | 0.7592 | 0.6528 |
| 53 | 0.0576 | 0.9443 | 0.9478 | 0.893 |
| 54 | 0.4054 | 0.7082 | 0.7052 | 0.4865 |
| 55 | 0.4771 | 0.6873 | 0.6677 | 0.399 |
| 56 | 0.2407 | 0.7987 | 0.7595 | 0.6459 |
| 57 | 0.1544 | 0.8798 | 0.8409 | 0.7465 |
| 58 | 0.3275 | 0.725 | 0.7468 | 0.5253 |
| 59 | 0.1516 | 0.8545 | 0.8546 | 0.7386 |
| 60 | 0.351 | 0.6709 | 0.7221 | 0.5068 |
| 61 | 0.0345 | 0.9578 | 0.9756 | 0.9344 |
| 62 | 0.1453 | 0.8613 | 0.8645 | 0.7473 |
| 63 | 0.128 | 0.8738 | 0.8708 | 0.7817 |
| 64 | 0.1988 | 0.8094 | 0.7977 | 0.6857 |
| 65 | 0.0588 | 0.946 | 0.938 | 0.8889 |
| 66 | 0.3401 | 0.7081 | 0.7792 | 0.5226 |
| 67 | 0.0671 | 0.9351 | 0.9309 | 0.8742 |
| 68 | 0.4942 | 0.4912 | 0.5402 | 0.3966 |
| 69 | 0.1928 | 0.8114 | 0.8048 | 0.6938 |
| 70 | 0.092 | 0.9022 | 0.915 | 0.8349 |
| 71 | 0.0432 | 0.9565 | 0.9571 | 0.9181 |
| 72 | 0.0639 | 0.9382 | 0.9348 | 0.8809 |
| 73 | 0.2002 | 0.8051 | 0.8405 | 0.6826 |
| 74 | 0.0895 | 0.9064 | 0.9245 | 0.8372 |
| 75 | 0.126 | 0.8708 | 0.9065 | 0.7825 |
| 76 | 0.3901 | 0.6276 | 0.6334 | 0.4908 |
| 77 | 0.2938 | 0.7087 | 0.7051 | 0.5618 |
| 78 | 0.1493 | 0.8911 | 0.8595 | 0.7542 |
| 79 | 0.1667 | 0.857 | 0.8663 | 0.7311 |
| 80 | 0.1312 | 0.8773 | 0.8638 | 0.7701 |
| All | 0.1911 | 0.8222 | 0.8271 | 0.7267 |

Figure 2: Detailed Segmentation Results

## Detailed Analysis

Figure 2 analyzes the detailed performance of the image segmentation algorithm for differentiating rotten, fruit, and background regions in a set of images.

The error rate ranges from a very low 0.022 (indicating excellent segmentation) to a high 0.6681 (signifying significant segmentation errors). This trend is mirrored in the precision, recall, and IoU values, which are generally high for most images but reach their lowest points in Image 16.

Examples: High Error Images: Image 16, 35, 47 (color mismatch, lighting issues)
Low Error Images: Image 2, 20, 21, 23, 38, 53, 67, 74 (clean background, good lighting, high color contrast)

**Analysis of High Error Images with Low Precision, Recall, and IoU: Images 16,35,47**

**Focus on Image 16 (0.6681, 0.3463, 0.3195,0.3125)**

Image 16: This image exhibits the highest error rate and the lowest precision, recall, and IoU values compared to all other images.

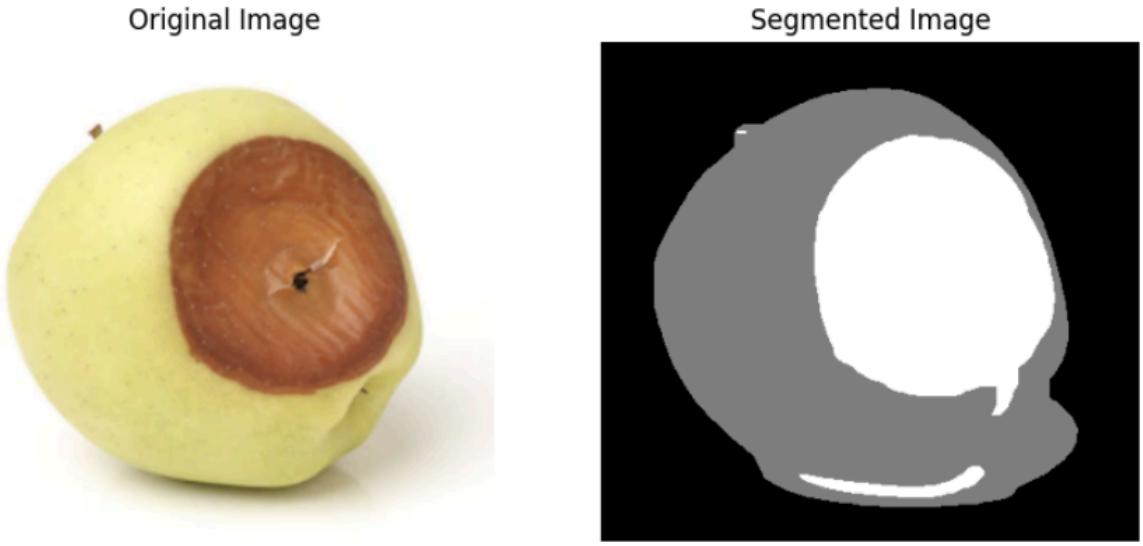


Figure 3: Image 16 before and after segmentation

Analysis of error:

From the figure, we observe a color clustering mismatch in the segmentation results.The segmentation algorithm incorrectly classifies the rotten part of the apple, which should be gray, as white, and the healthy part, which should be white, as gray. Additionally, the presence of shadows on the apples introduces further misclassification by rendering parts of the apples gray, which is inaccurate.

**Image 35 ( 0.6254,  0.3914, 0.3739, 0.3401)**

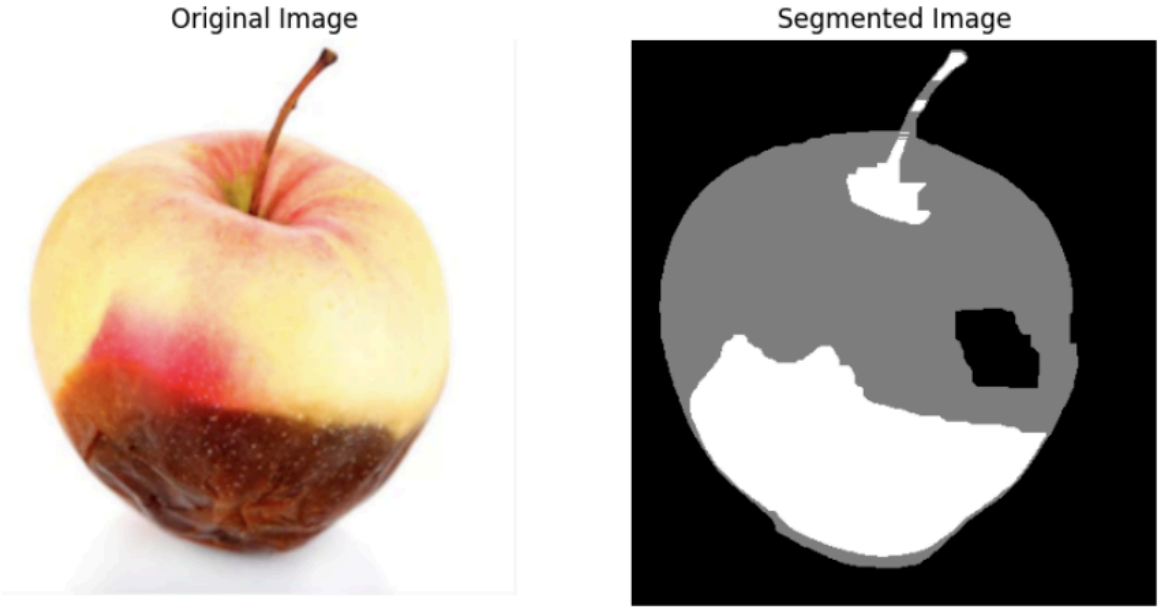


Figure 4: Image 35 before and after segmentation

Analysis of error:

Similar to image 16, color mismatch leads to incorrect segmentation. The segmentation algorithm incorrectly classifies the rotten part of the apple, which should be gray, as white, and the healthy part, which should be white, as gray. Additionally, fruit and background regions are confused, causing some fruit areas to be classified as background. Lighting might also play a role in this misclassification.

**Image 47 : (0.5733)**

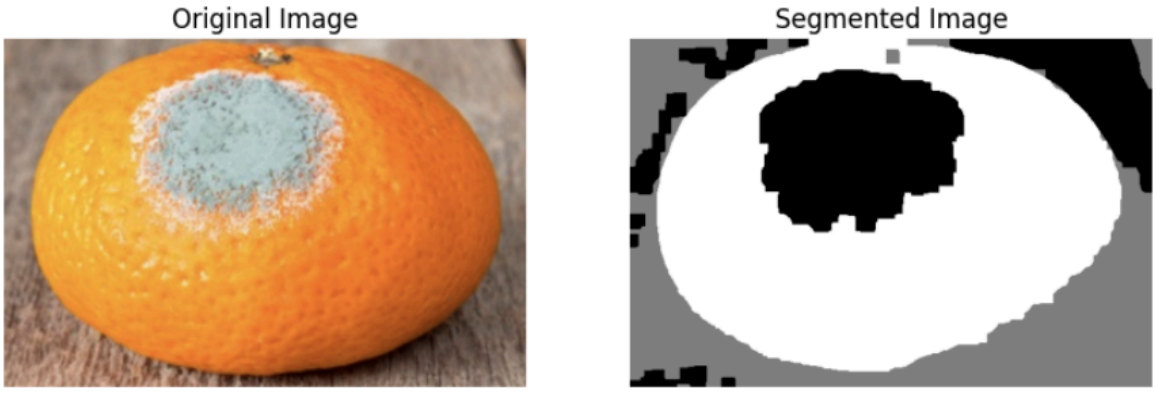


Figure 5: Image 47 before and after segmentation

Analysis of error:

Here, the background color itself seems to be causing issues. The algorithm misinterprets the background color as rotten and rotten part as background, leading to a segmentation error.

In conclusion, from figures 3 and 4, we noticed that the causes of mismatching are due to lighting variations and the fact that both apples are white, resulting in a relatively low R-B difference. Additionally, figure 5 shows that the background is a significant issue.

**Analysis of Low Error Images with High Precision, Recall, and IoU: Images 20,10,71**

**Image 20: 0.022,0 0.9796, 0.977, 0.9572**

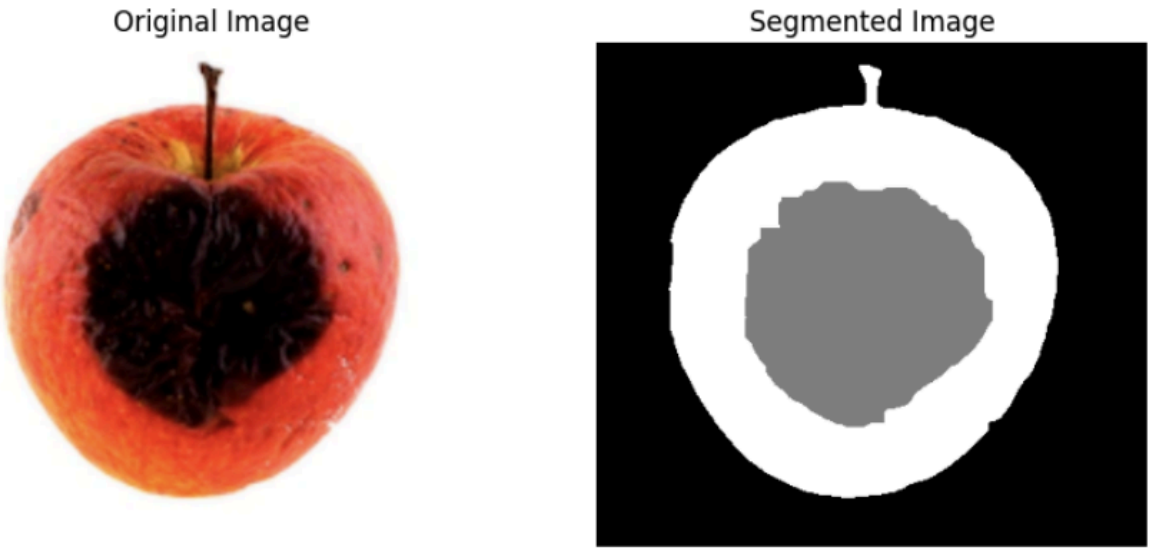


Figure 6: Image 20 before and after segmentation

From the graph, we can see that the apple matched well.

**Image 10: 0.0301, 0.9699, 0.97, 0.9417**

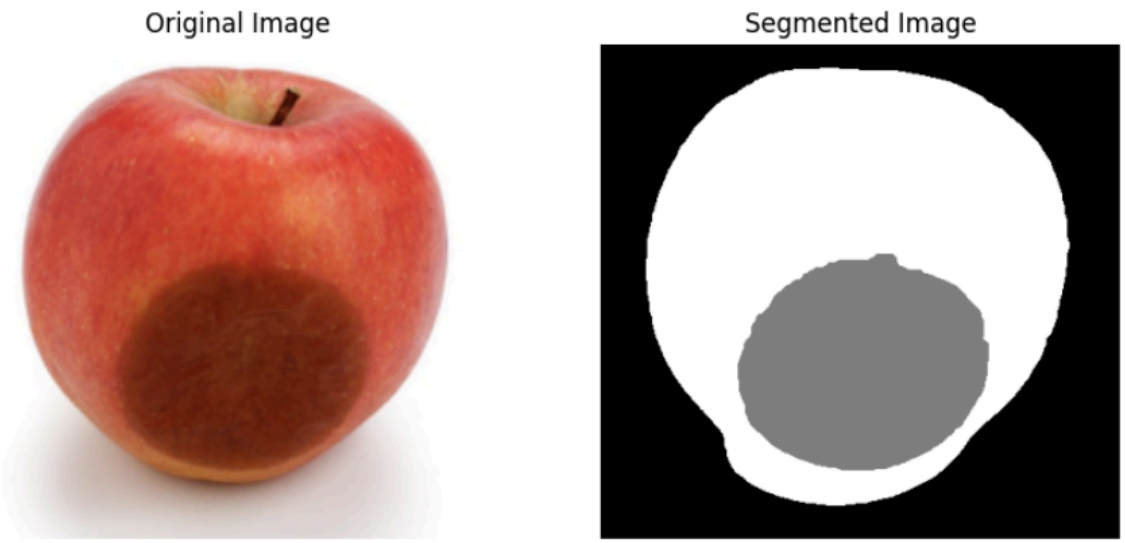


Figure 7: Image 10 before and after segmentation

From the graph, we can see that the apple matched well.

**Image 71:  0.0432, 0.9565, 0.9571, 0.9181**

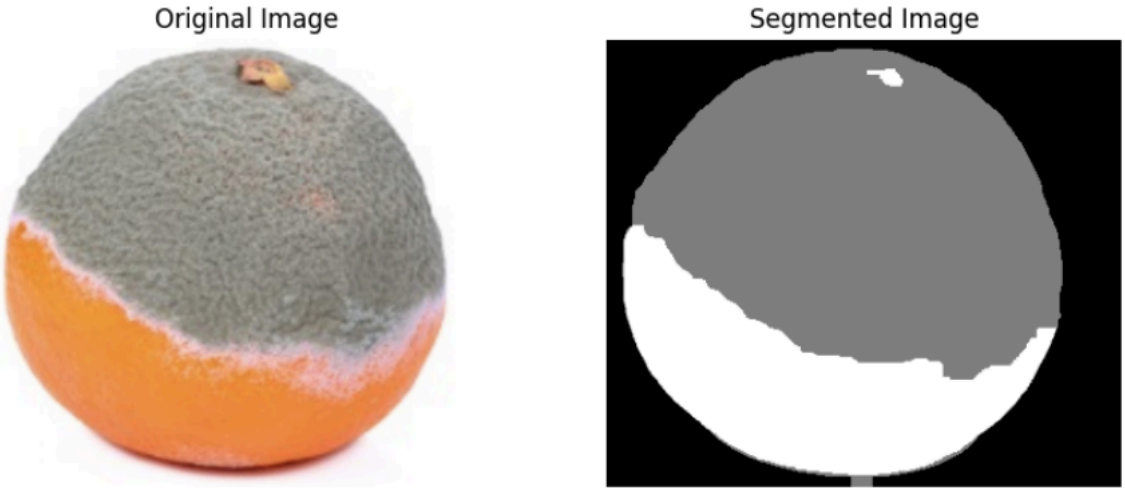


Figure 8: Image 71 before and after segmentation

From the graph, we can see that the orange matched well.

Overall, these images achieve high-performing images because they mostly have clean backgrounds, uniform lighting and distinct coloration.

## Suggestions for Improvement

The analysis revealed that lighting variations, background color inconsistencies, and difficulties with white apples contribute to segmentation errors. To mitigate these limitations, several approaches can be explored. Pre-processing the images to normalize or adjust the background color can minimize its impact on segmenting rotten and fruit regions. Additionally, specific pre-processing steps tailored for white apples, such as color normalization or background subtraction, could be investigated to improve their segmentation accuracy. Furthermore, incorporating features beyond color information, like texture or spatial data, could offer a more robust representation for differentiating rotten and fruit regions during segmentation. Machine learning techniques specifically designed for fruit segmentation are also promising. By training a model on a diverse dataset, it could learn to handle challenging cases like white apples and varying lighting conditions. Finally, while RGB provided the best results so far, exploring alternative color features like LAB, grayscale, or HSV could be beneficial. Further experimentation can help identify the optimal color representation for this task. By implementing these suggestions, we can potentially enhance the segmentation accuracy, especially for images with the identified challenges.

# Reference

K-means clustering

https://www.kaggle.com/code/abrahamanderson/image-segmentation-with-k-means-clustering