Fruit Inspection IPCV Project

Freddy Fernandes
freddy.fernandes@studio.unibo.it

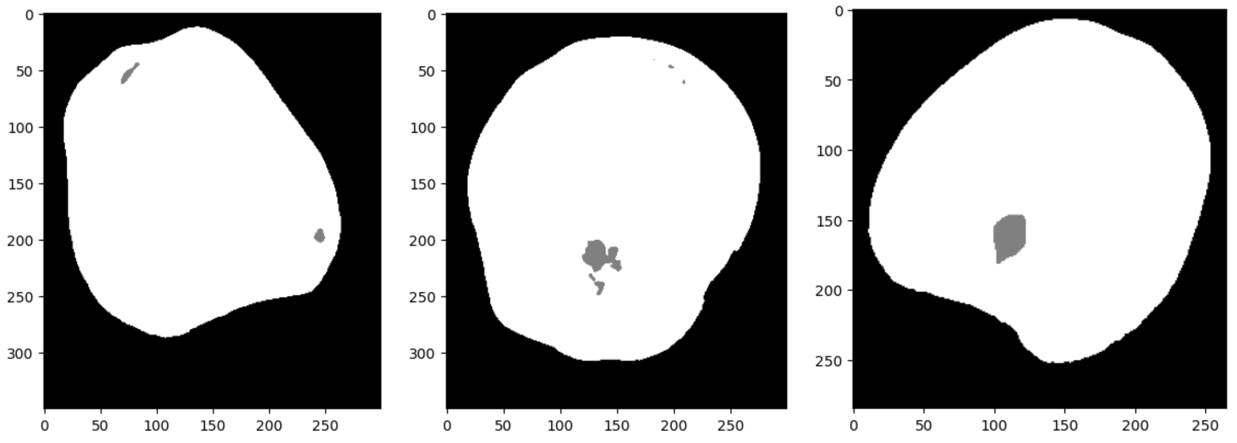
1. Introduction

This project focuses on Fruit inspection. We are provided with images of Apples, both in Near Infra-Red and colour. These apples in the images have some defects such as bumps or holes, or reddish-brown russet regions. The project involves two tasks. Firstly, it aims to develop a robust algorithm for detecting defects in images of apples. Then for the second task, a Mahalanobis distance function to detect russet defects in apple images across different color spaces.

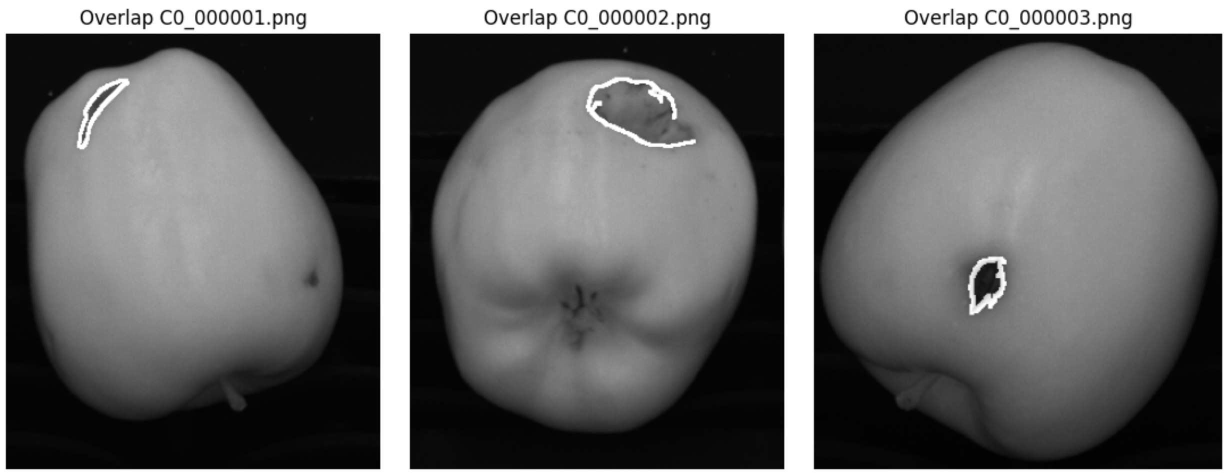
2. Task A: Fruit Segmentation and Defect Detection

In the first stage of our project, we aim to identify external defects in images of apples. These defects range from holes to indentations on the surface of the apples. Firstly I used the grayscale histogram approach to identify any particular peaks in the histogram levels. I applied a Gaussian filter to try and smoothen some noisy values. The result from the histograms was not very interesting. I tried using a manually set threshold value and using the binary threshold function to separate the foreground/background. But the manual threshold values were not accurate enough to properly binarize the image. Then I also used the Otsu algorithm threshold function which generates an optimal threshold value for binarization, which resulted in better images with most of the defects being binarized as the black background. I used the flood fill function to fill these holes in the image. I then proceeded to try Sobel edge detector and Canny edge detector to identify the edges in the images. Canny edge detection was very good at detecting the edges. This includes the complete outside edges as well as the edges of the defects in the apple image. Then using the mask computed earlier, I used erosion to disconnect the apple's outer edges and then dilation to highlight the inner defect edges. So, I was able to mark the edges of only the defects and save these. Finally, I overlapped these edges onto an actual image of the apple to show the defects being correctly marked on the apple. I performed the same pipeline process for all the three apple images.

Results:
I filled in the defect holes for the three images. The holes were correctly filled for images 1 and 3 but in image 2, the natural hole at the bottom of the apple was filled as this was the binary mask I got with Otsu's algorithm.



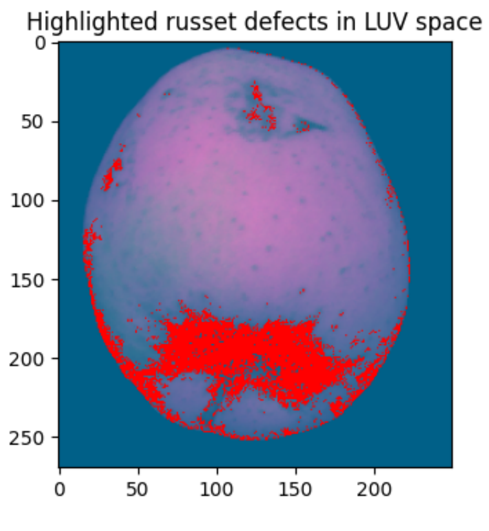
I was able to detect the major defect marks in the apples. But in image 1, I could not mark the circular defect in the side as adjusting the threshold value to include this was causing some false positives with marking the natural holes at the base of the apples too.

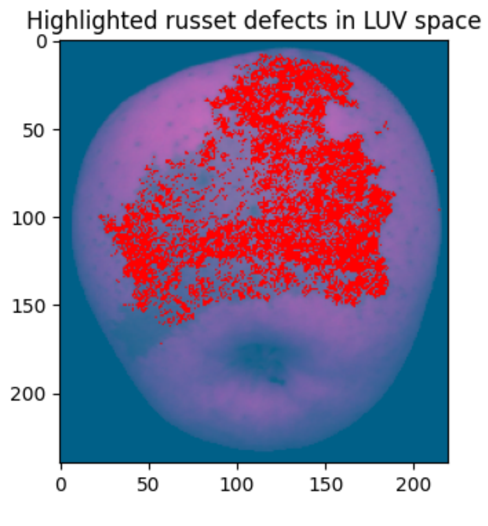


3. Task B

This task involved detecting some reddish-borwn regions in the apple images, called russet defects. Similar to the first task, I used the grayscale histogram approach to identify any particular peaks in the histogram levels. I applied a Gaussian filter to try and smoothen some noisey values. I used a manually set threshold value and using the binary threshold function to separate the image. This is useful for later to be used as a mask. Then I also used the Otsu algorithm threshold function. This gave me some weird binarized results. I used the flood fill function to fill these holes in the binary threshold images and use this as a mask for the color spaces. Then I define the new color spaces HSV, HLS, LUV and convert each image to these spaces. Then I am computing a small region of interest (in the russet region of the image) and computing the mean for each channel in the color spaces. I then reshape the images into 2D arrays and compute the (inverse) covariance matrix in each image. Then through the images, I normalize the distances between [0,255] and mark pixels thusing the scipy spatial mahalanobis distance function, I am computing the mahalanobis distance for each pixel (to the mean of the ROI) in each of the images. Then I am reshaping the distances to match the original image dimension. Then looping through the image i am marking the pixels where the mahalanobis distance is lower than a defined threshold. I only found the LUV color space to be useful and the russets are marked rather well. I had to manually assign the roi in each image depending on the russet location. It is not perfect as it marks some wrong parts too.

Results:
I was able to detect the russet region in image 2 with a good result. In image 1, i am not able to detect the upper russet region and there is also some regions marked on the edges.





4. References:

- Image Thresholding
- OpenCV: Smoothing Images
- Edge Detection Using OpenCV | LearnOpenCV #
- Adding (blending) two images using OpenCV
- Changing Colorspaces
- numpy.cov — NumPy v1.26 Manual
- numpy.linalg.inv
- scipy.spatial.distance.mahalanobis
- Mahalanobis-distance-example