Project work IP&CV

Farinola Francesco - francesco.farinola@studio.unibo.it
MAT. 954302

First task

Outline the image with a binary mask and search for defects

1st step - Preprocess image:

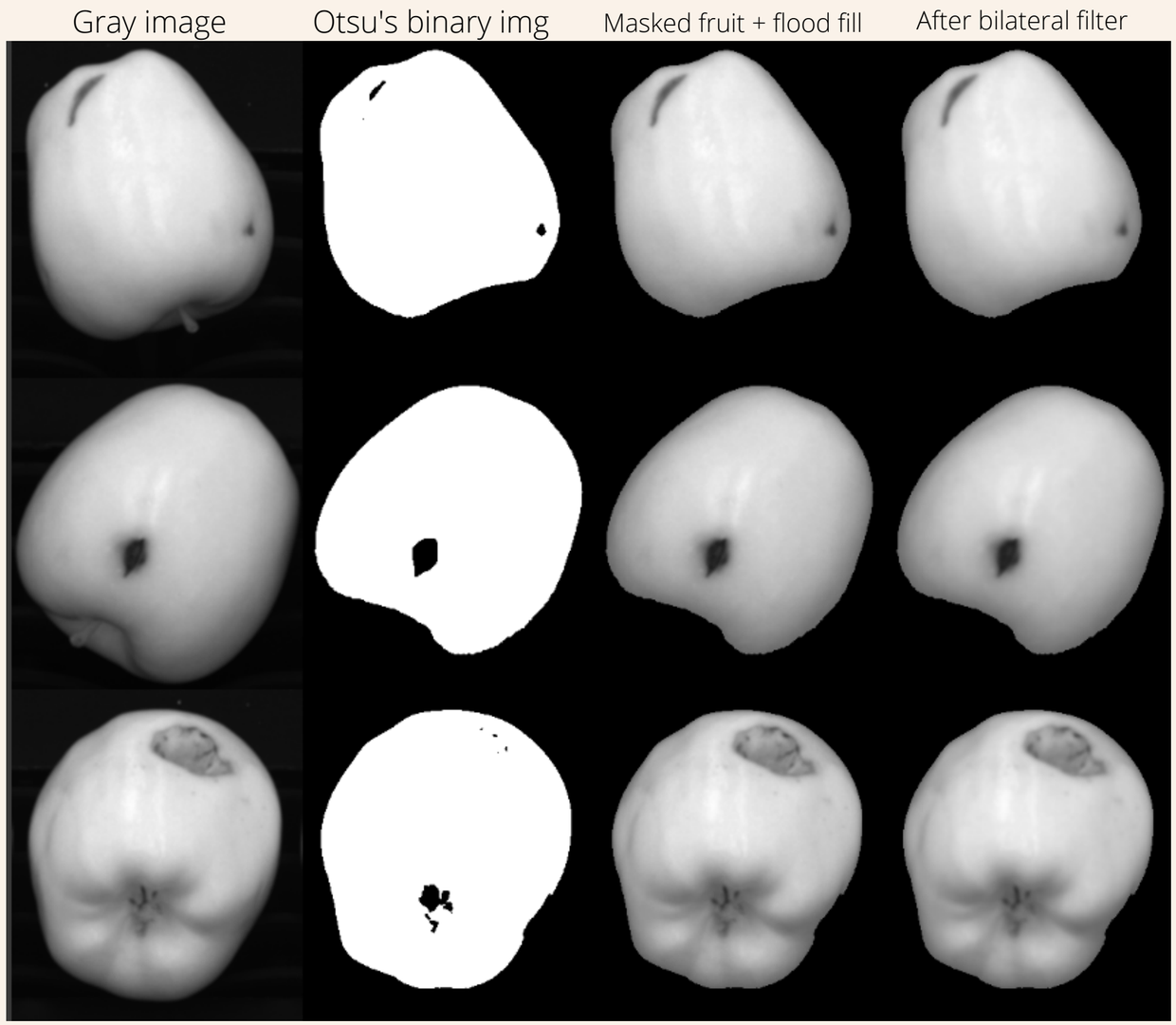
When computing a binary image, we are supposed to have a bi-modal gray-level histogram but since images are noisy we need to smooth the signal first:

- Apply a **Gaussian filter** to the gray image
- Compute a binary mask using the **Otsu's method**
- Apply **flood fill** to the binary mask to fill holes of the fruit region in order to get a mask of the fruit
- Apply the mask to the gray image and smooth signal via a **Bilateral filter** to preserve edges, so we can find edges of defects later

Parameters

- **Gaussian filter** : 3x3 kernel with sigma computed automatically by OpenCV using the rule of thumb
- **Otsu's method** applied via *cv2.threshold(v, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)*
- **Flood fill** is applied via *cv2.floodFill* by giving as parameters the binary mask and a mask with all 0 values
- **Bilateral filter**: d (diameter) = 5 - small kernel for fast computation and both SIgmas = 40. Like this, pixels closer and with similar intensity will have bigger weights

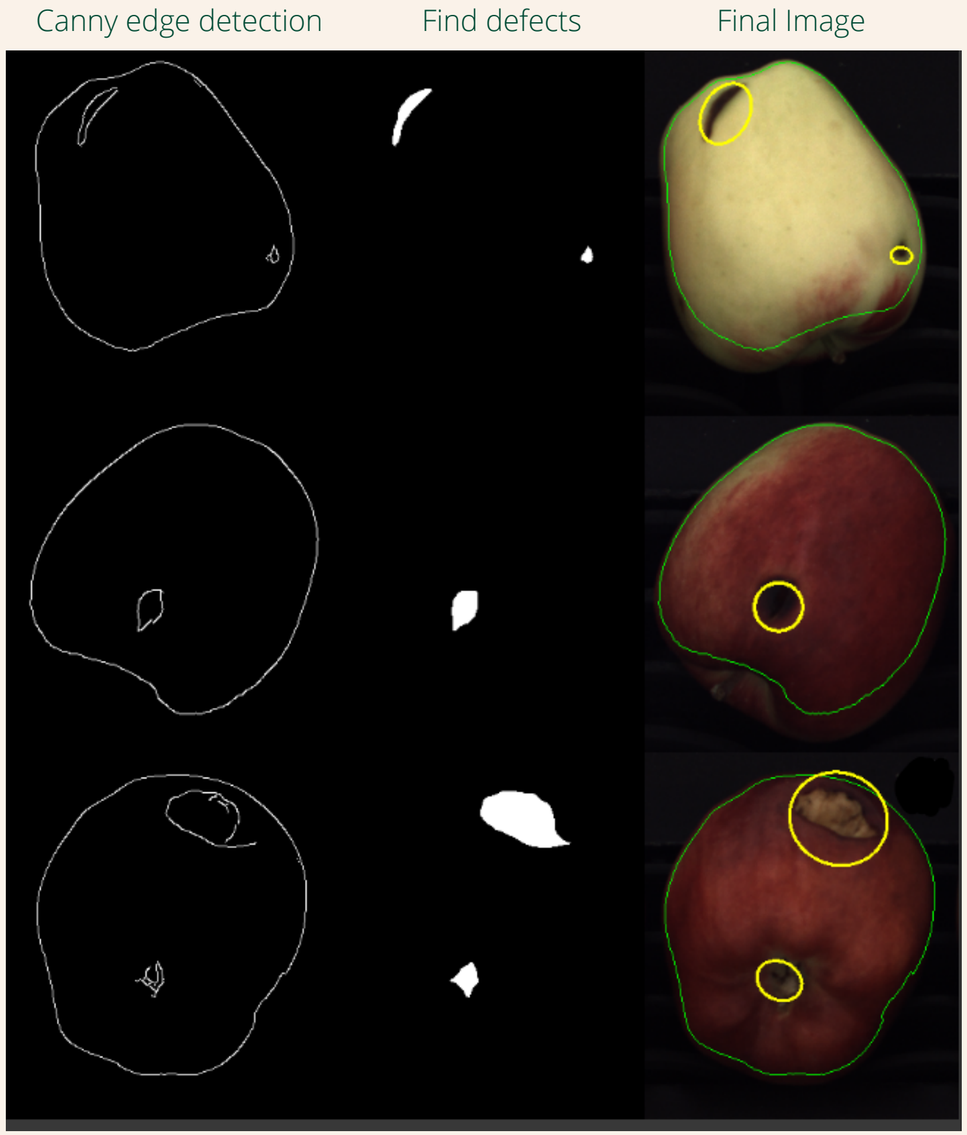
The goal of this first step is to identify a mask for the fruit, and not having defects already detected.



First task

2nd step - Detect defects

- Detect edges inside the fruit via a **Canny Edge detector** with hysteresis thresholds: *T_low = 0* and *T_high = 130*. Like this all pixels will be weak edges while edge pixels whose gradient is higher than 130 will be strong ones.
- To find defects edges:
  - Create a **background mask** : 255 - mask
  - **Dilate** the latter mask via a 5x5 dilation x 3 iterations
  - **Subtract** this mask to the canny's one to get fruit/defects contours
- Consolidate edges via a **closing** operation (dilation + erosion) with an elliptic structuring element - like this we will fill holes with ellipses of the defects
- Finally, we **draw** ellipses on the color image over contours (cv2.findContours) whose area is larger than a minimum of 10



Second task

Identify the russet or at least some part of it - possibly with no False Positives

Two approaches:

Preprocess image

- Compute a binary mask of the B/W image
- Get connected components from binary mask and get the one with maximum area = fruit
- Apply flood fill like in first task
- Erode contours of fruit with a 5x5 structuring element 3 times to darker shades of the color, due to shadows

1. Detection of russet via **K-means** color clustering

Convert the image to the **CIE Lab** color space since it is the nearest to human's eye perception of colors.
Compute K-means to clusterize pixel colors and then sample russet colors from the two images so we can get the centroid nearest to those samples as russet indexes.

2. **Adaptive thresholding** meaningful color channels

Convert the two images to different color spaces: **HLS, HSV, Lab, Luv**
Plot each image as a **gray image** using values from **single channels** so as we can spot which color channel models better russets.
Then, apply adaptive thresholding on the smoothed gray image of most meaningful channel.

Second task

K-Means

Before applying K-means, we smooth the image via a **Bilateral Filter**, then **convert** it to the Lab color space and **normalize** values in a [0,1] range.
Since a fruit may have different shades of colors we cannot set a fixed number of clusters. To get the best K, we perform iterative k-means with k spanning from 2 to 7. For each iteration, we compute a performance index which is called **scaled inertia** and is computed as:

$$Scaled\ Inertia = \frac{Inertia(K)}{Inertia(K=1)} + \alpha \cdot K$$

So, we get the k with **minimum** scaled inertia and get labels and centroids for an image.
After this we **sample** from the given images **two shades of russet** color : light and dark brown and compute the **city-block distance** of centroids from these samples so we take as russet index the centroid with minimum distance.
Finally, we get as defect mask, all the pixels whose label from k-means has been assigned to the russet index = centroid label

Second task

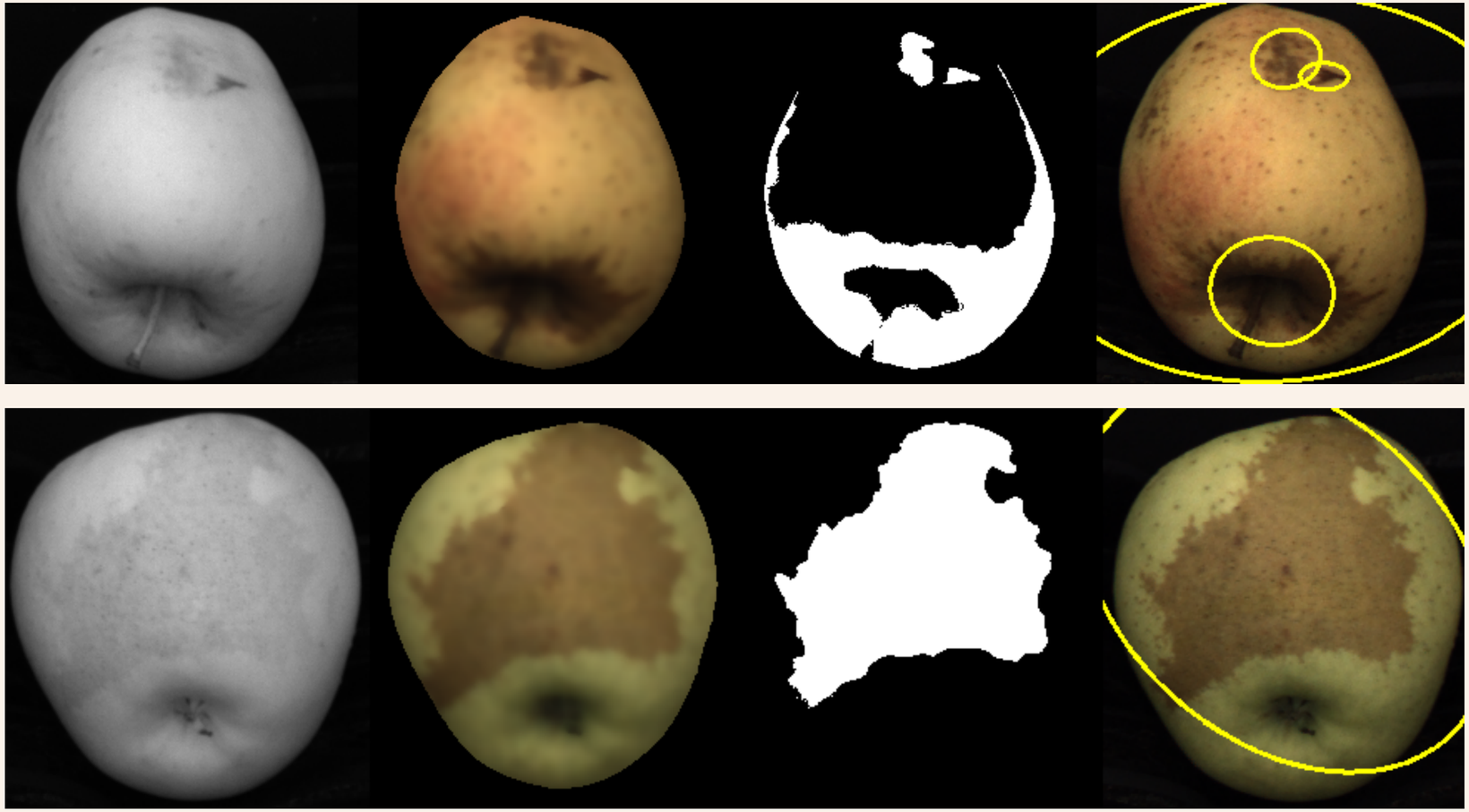
K-Means Result

Gray image masked

Eroded/blurred color image

Russet mask

Final image



Second task

Adaptive thresholding meaningful color channels

With matplotlib we are able to plot single channels intensity values as a gray image in order to spot which one enhances most the russet color. From the images below we can clearly see that the **V channel of Luv** color space give more contrast to the russet.

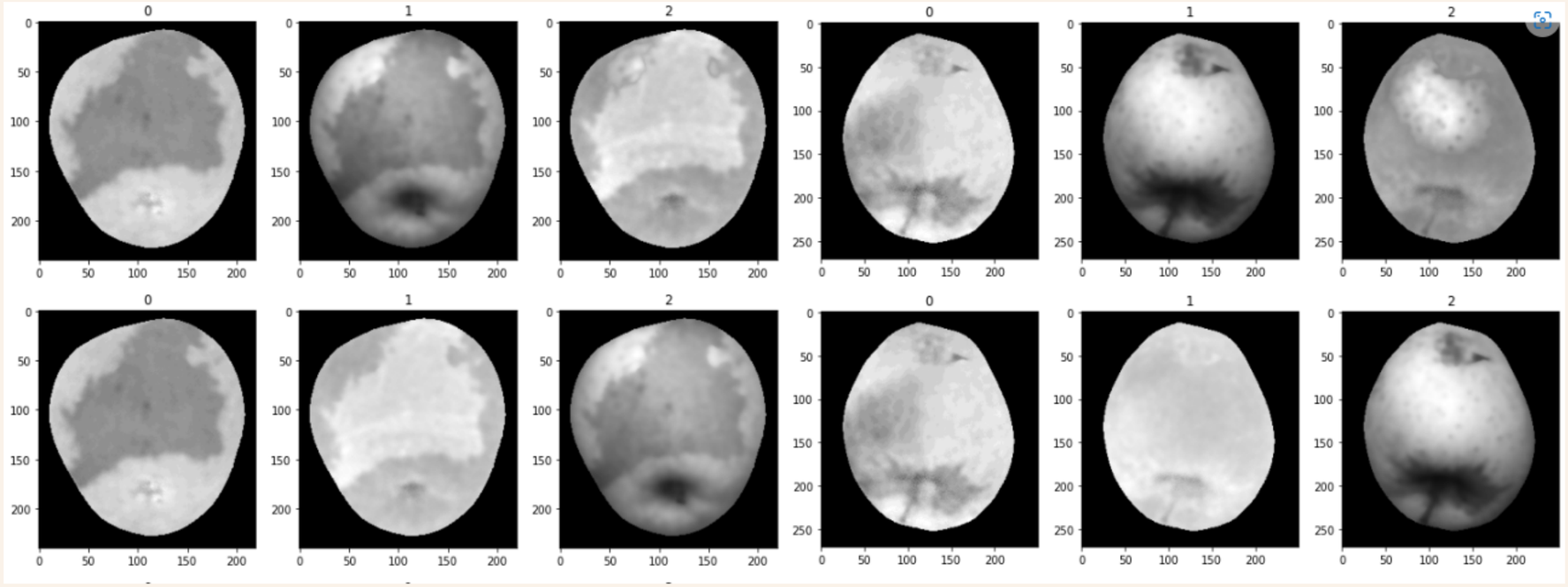
First image

Second image

Color space

HLS

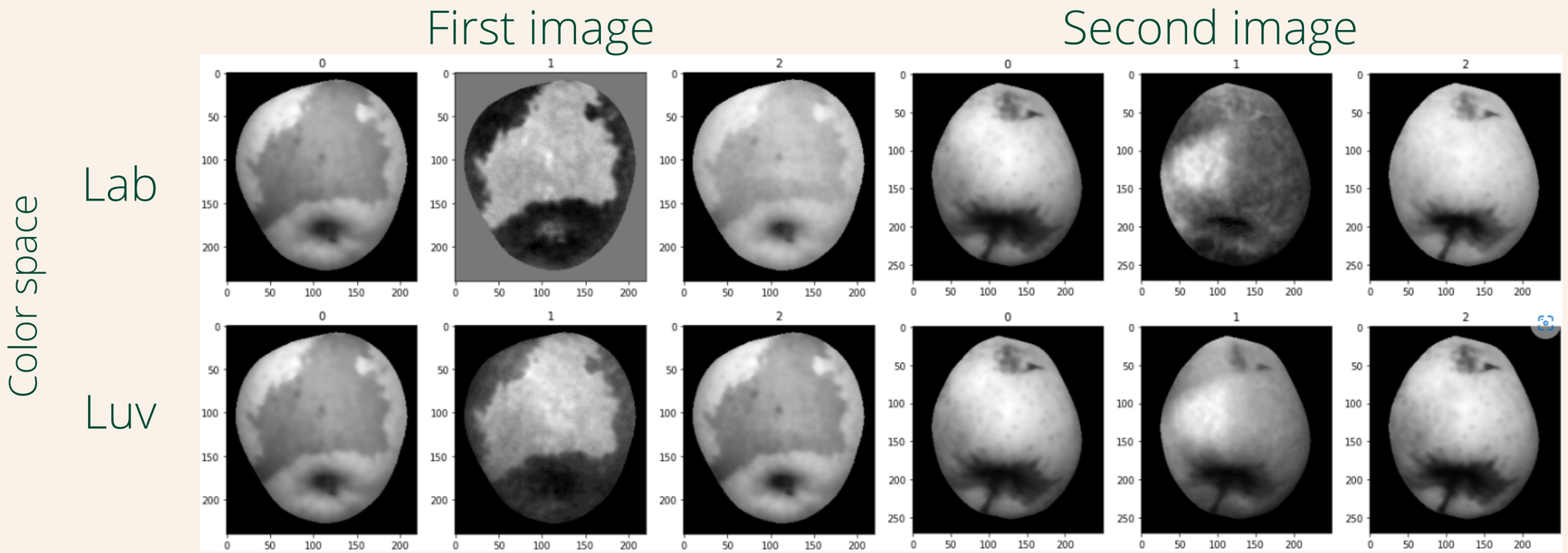
HSV



0,1,2 represent respectively the first, second and third channel of the correspondent color space

Second task

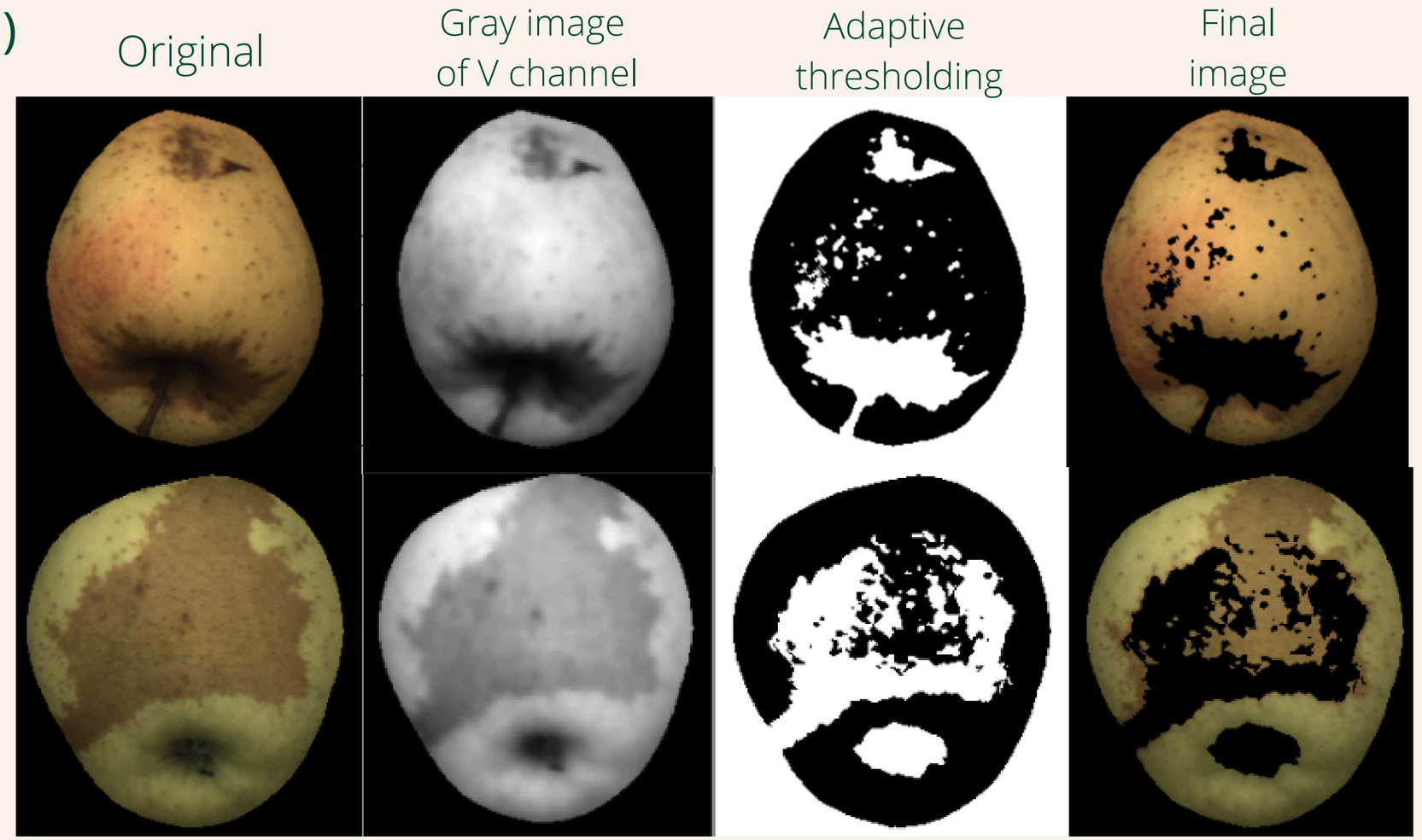
After extracting the V values, from the Luv image, we apply a **Gaussian filter** to smooth signal and apply **adaptive threshold** to be robust to light changes using a *Gaussian like mean* and using a *block size of 145* and constant *c=0*. After applying adaptive threshold we end up with a binary mask of the russet.



0,1,2 represent respectively the first, second and third channel of the correspondent color space

Second task

Adaptive thresholding meaningful color channels - Result (mask of V channel in Luv space)



Third task

Kiwi inspection - Detect defects with special care for background noise

This task can be addressed like the first one. To get the kiwi's mask:

- The gray image is thresholded in order to get a binary image - this time with a **fixed thresh = 40**
- **Flood fill** the binary image
- Apply an **Erosion with a 7x7** structuring element to separate potential background noise from contours of the kiwi
- Compute the **connected components** and get the one with **maximum area** to get the kiwi.

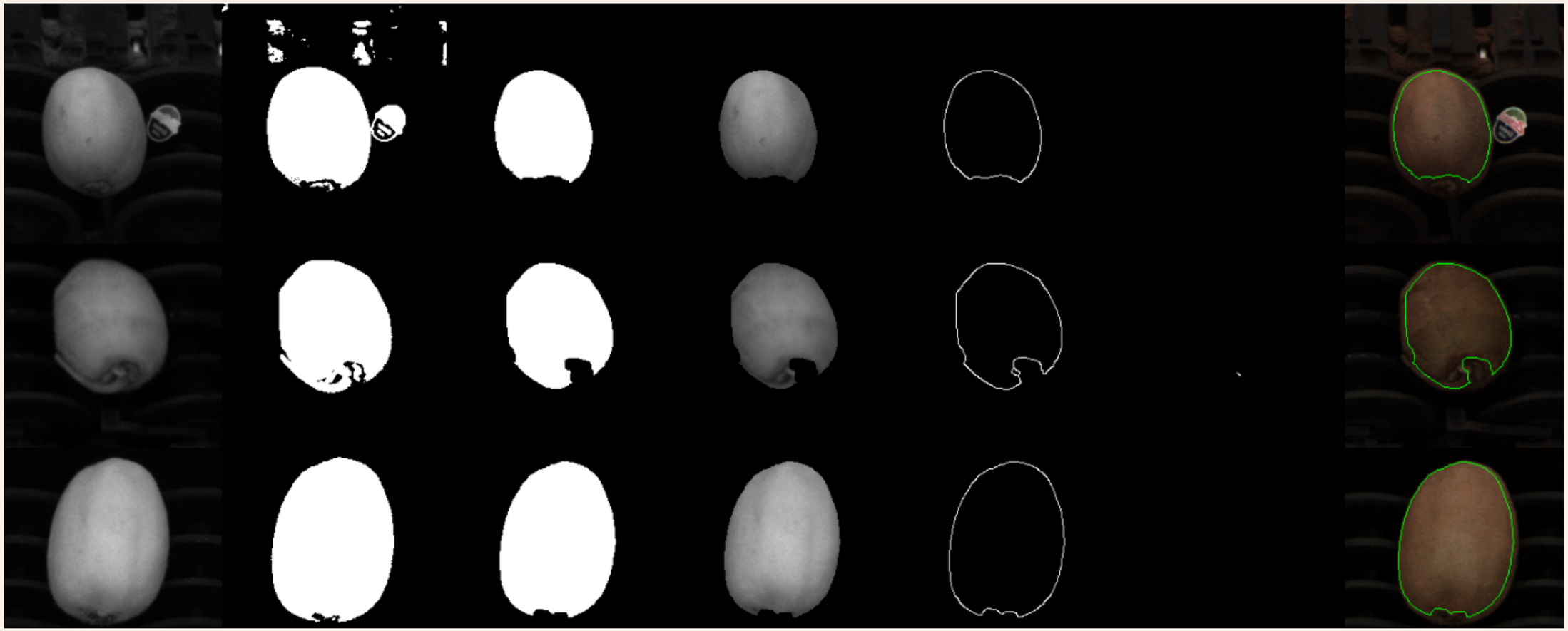
To detect defects:

- Apply a **Bilateral filter** to the masked image before computing edges (7, 40, 50)
- Perform **Canny's edge detection** but fine-tuning hysteresis thresholds - Tlow = 10, Tupper = 110
- Compute the *background mask* (255 - mask) and *dilate* it. then **subtract** this to canny's mask
- Consolidate defects with a **closing operation** with elliptic structuring element

Third task

Kiwi inspection - Result (1)

Original | Binary Mask | Flood Fill + Erosion + Max area C | Bilateral filter | Canny's edges | Defect mask | Final



Third task

Kiwi inspection - Result (2)